A multi-channel front-end for synthetic aperture sonar

Blair Bonnett, B.E. (Hons)

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

March 2010

ABSTRACT

Synthetic aperture sonar (SAS) is a wide-beam sonar technique commonly used for mapping the seafloor at high resolution. The Acoustics Research Group at the University of Canterbury operates a towed SAS system known as KiwiSAS-IV. This is currently being redesigned with the aim of reducing the weight, size and power requirements of the system. The long term goal is to make it capable of being mounted on an autonomous underwater vehicle (AUV) so that mapping of remote and/or dangerous waters can be accomplished without human interaction.

This thesis presents the design of the front-end electronics used to drive the 36 transducers to produce the acoustic beam and receive the returning signals after they have reflected off a target. To achieve sufficient range, the transducers are driven with a $200\,V_{p-p}$ signal with a maximum frequency of 110 kHz. This design uses class D switching amplifiers to generate these waveforms. The AD9271 integrated circuit, which can handle eight transducers simultaneously, is used to amplify the incoming signals and sample them at up to 50 MHz. This high sampling rate multiplied by all 36 transducers results in an amount of data which is too great for a conventional microprocessor-based system to handle. Instead, an FPGA is used to receive this data, decimate it using multiplier-free cascaded integrator-comb (CIC) filters, and then pass it to the back-end system for further processing and storage.

A prototype circuit was created to test the theory developed in this thesis. This showed that the system is capable of generating the necessary waveforms and amplifying, capturing, and decimating the returning signals. However, further refinement is required before it is able to be used in the sonar system.

ACKNOWLEDGEMENTS

First of all I would like to thank my supervisors, Dr Michael Hayes and Professor Peter Gough, for all the help and guidance they have provided over the course of this research. Michael in particular has handled the assorted questions I threw at him with aplomb. Without his expertise ranging from sonar signal processing to the intricacies of embedded system design, the progress achieved in this thesis would be significantly less. Thanks are also due to Mike Cusdin for his help with the assorted quirks of Altium Designer and for providing information on the existing sonar system.

To those who have been lucky/unfortunate enough to share an office with me, thank you for providing helpful advice and light relief over the past couple of years.

To my family: thank you for all the love, support and encouragement you have shown over the past 23-and-a-bit years. I could not have achieved this without it.

And, last but most certainly not least, to Deborah. Thank you for standing by me throughout this journey. Words cannot say how much it means.

CONTENTS

LIST OF FIGURES

ACRONYMS

ADC analogue to digital converter, a circuit which coverts continuous (analogue) signals to their discrete (digital) equivalents.

AUV autonomous underwater vehicle, an robotic system which operates underwater without the need for human control. Also known as an unmanned underwater vehicle (UUV).

AWGN additive white Gaussian noise, a noise model which assumes all the noise is additive and has a flat spectrum and Gaussian amplitude distribution.

BEM boundary element method, a numerical technique used to solve partial differential equations which have been expressed in boundary integral form.

BIBO bounded-input, bounded-output stability, a stability condition for a system which guarantees that a bounded (i.e., always finite) input signal results in a bounded output.

BVD Butterworth-Van Dyke model, a model of the electrical admittance of a general capacitive electromechanical system. It consists of a bulk capacitor in parallel with one or more RLC branches modelling the resonances of the system.

CIC cascaded integrator-comb filter, a hardware efficient decimation or interpolation filter requiring no multiplication. The decimation version consists of a series of integrators followed by a series of comb filters; for interpolation the order is switched.

DAC digital to analogue converter, a circuit which converts discrete (digital) signals to their continuous (analogue) equivalent.

Δ-Σ delta-sigma, an ADC architecture which samples the signal at high speed but with limited resolution, followed by a decimator to reduce the sampling rate and increase the resolution.

DDR double data rate, a transfer mode where data is output on both the rising and falling edges of a clock.

DFT discrete Fourier transform, a version of the Fourier transform which is discrete in both the time and frequency domains.

DTFT discrete-time Fourier transform, a version of the Fourier transform which is discrete in the time domain but continuous in the frequency domain.

ENOB effective number of bits, the maximum number of bits of resolution that can be extracted from a sampled signal with a certain SINAD.

FEM finite element method, a numerical technique used to solve differential equations.

FET field effect transistor, a type of transistor which uses electric fields to control its conductivity.

FIR finite impulse response filter, a digital filter which has an impulse response which settles to zero within a finite number of sampling intervals.

FPGA field programmable gate array, a programmable IC which contains a large number of logic blocks and interconnects to wire the blocks together to implement the desired operation.

IC integrated circuit, a miniaturised circuit manufactured on a piece of semiconductor material.

IIR infinite impulse response filter, a digital filter which has an impulse response which is non-zero for an infinite length of time.

LNA low noise amplifier, an electronic amplifier designed to amplify very weak signals without adding significant noise. Usually located as close as possible to the detection device.

LVDS low-voltage differential signalling, a differential signalling system designed to run at high speeds. It uses a common-mode voltage of 1.25 V with a differential voltage of around 350 mV. It is capable of speeds up to at least 3 Gbps.

PCB printed circuit board

PVDF polyvinylidene fluoride, a thermoplastic fluropolymer which can be used as a piezoelectric element.

PSD power spectral density, a measure of how the power in a system varies as a function of frequency.

PWM pulse-width modulation, a method of modulating a square wave by varying the percentage of time it is on.

PZT lead zirconate titanate, a piezoelectric ceramic consisting of lead, zirconium, titanium and oxygen commonly used in acoustic transducers. The chemical formula is $\mathrm{Pb}[\mathrm{Zr}_x\mathrm{Ti}_{1-x}]\mathrm{O}_3$ where $0 < x < 1$ varies depending on the manufacturer and application; a commonly used value is $x = 0.52$.

RAM random access memory, volatile memory that can be read from and written to at very high speed.

SAS synthetic aperture sonar, a widebeam sonar system which insonifies each portion of the seafloor multiple times and uses this information to synthesise an aperture much larger than the one used.

SINAD signal-to-noise-and-distortion ratio, the ratio of the power in the desired signal to the level of noise and distortion present. Typically measured in dB.

SNR signal-to-noise ratio, the ratio of the power of the desired signal to the noise power present. Commonly given in dB.

SPI serial peripheral interface, a serial data transfer protocol.

T/R transmit/receive switch, a circuit which is used to switch a system between transmit mode and receive mode, often automatically. Commonly used to protect a sensitive receive circuit from a transmitter.

VGA variable gain amplifier, a type of amplifier with a gain that varies in response to a control voltage.

VHDL very high speed integrated circuit hardware description language, a language designed to describe the operation of logic circuits and commonly used to program FPGAs.

CHAPTER 1

INTRODUCTION

Begin at the beginning and go on till you come to the end: then stop.

Lewis Carroll
Alice's Adventures in Wonderland

CHAPTER 1

INTRODUCTION

Imaging the seafloor using optical methods is generally unfeasible except over short distances as the high conductivity of water hinders the propagation of electromagnetic waves [Hunter 2006]. Additionally, sediment such as mud and sand can often be held in suspension within the water, further degrading the viability of optical imaging. However, the high density of water allows acoustic waves to travel with minimal loss, although low frequency signals (typically less than 500 kHz) are used as attenuation increases with frequency [Urick 1975]. As a result, sonar imaging is the most common method of imaging the seafloor [Medwin and Clay 1998].

Sonar is a time-of-flight technique. A pulse of acoustic energy — commonly known as a ping — is transmitted and the time it takes to reflect back off a target is measured, from which the range to the target can be inferred. If an array of receivers is employed, the range measured by each element in the array will differ; simple geometry can then be used to find the direction of the target.

1.1 Side-looking sonar

A side-looking (or side-scan) sonar is a type of imaging sonar which is moved along the edge of the scene (or area of interest). The direction of motion is referred to as the along-track direction, while the energy is transmitted in the perpendicular or across-track direction. The resolution, or minimum distance two targets must be separated by in order to be individually detected, in the across-track direction δ_r is determined by the bandwidth of the transmitted signal, while the along-track resolution δ_a depends on the spatial bandwidth of the system [Hayes and Gough 1992].

1.1.1 Narrow-beam sonar

As the name implies, a narrow-beam sonar focuses the acoustic energy into a tight beam so that only a small portion of the seafloor is insonified in each ping. As illustrated in Figure 1.1a, the system is moved across the desired scene to image it strip by strip; the results can then be combined to form a single image of the area. The along-track resolution is limited by the width of the beam, and can be approximated by

$$\delta_a \approx \frac{Rc}{fD}, \tag{1.1}$$

where R is the range, c the speed of sound[1], f the frequency of the signal and D the aperture width [Caprais and Guyonic 1997; Hunter 2006]. As low frequency signals are used to minimise attenuation, this leads to impractically large aperture widths to obtain high-resolution images. For example, if the transmitted frequency is 100 kHz and a 2 cm resolution is desired at a range of 200 m, (1.1) gives the required aperture width as 150 m. Additionally, as the range increases, the resolution worsens.

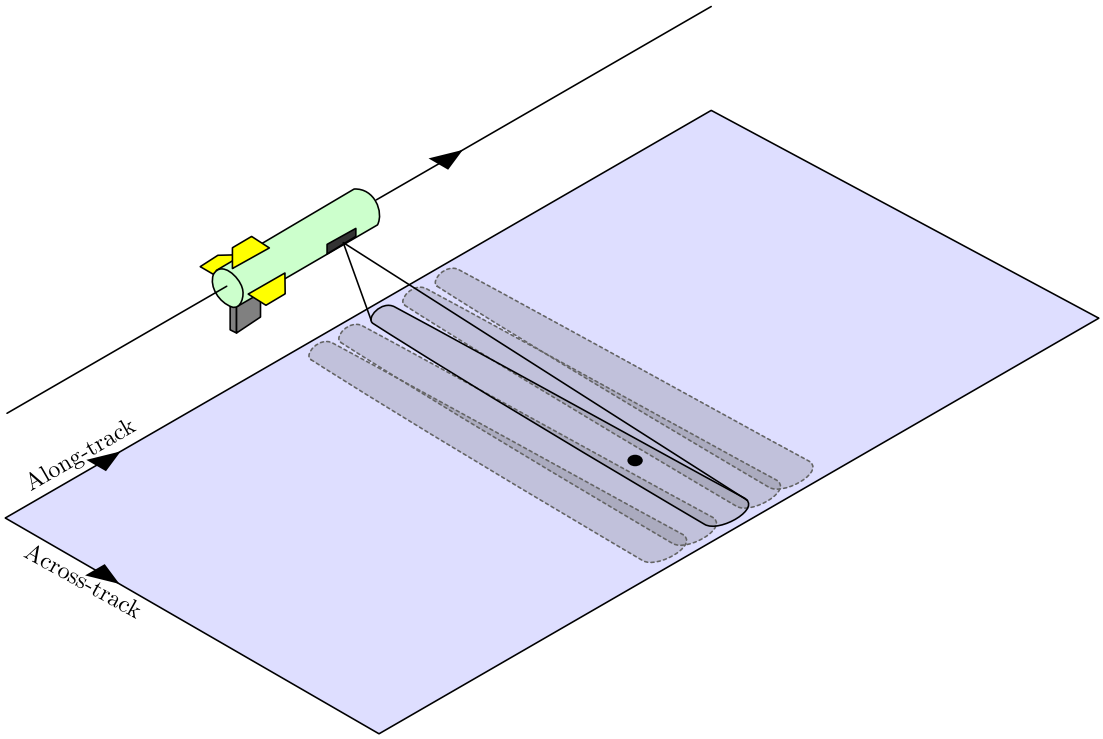
1.1.2 Synthetic aperture sonar

By contrast, synthetic aperture sonar (SAS) is a wide-beam sonar system, employing a relatively narrow aperture to generate a wide acoustic beam. As shown in Figure 1.1b, this means that each target will appear in the return of multiple pings. If the pings exhibit phase coherence, they can be combined to form a single image of the scene, effectively synthesising a large aperture [Hayes and Gough 1992]. As the beam spreads with increasing range, the more distant a target is the more pings it will appear in. This extra information allows the along-track resolution of the reconstructed image to be independent of range [Hawkins 1996]. The along-track resolution is given by

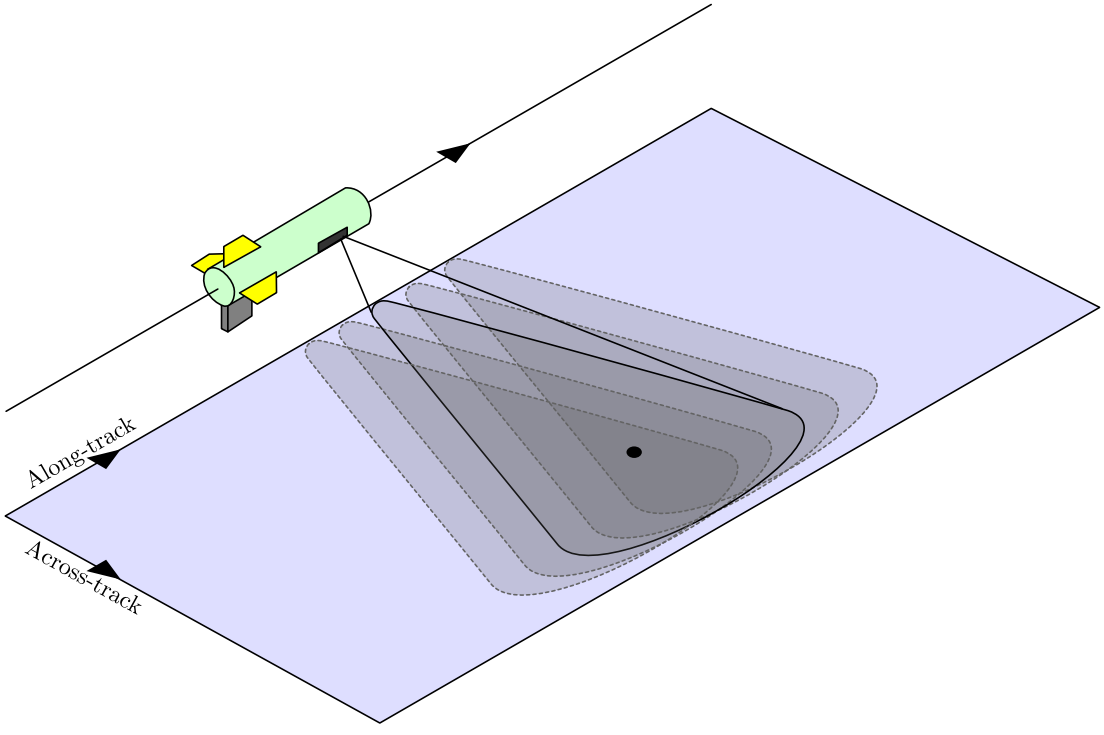
$$\delta_a = \frac{D}{2}, \tag{1.2}$$

i.e., it is also independent of the transmitted frequency [Caprais and Guyonic 1997; Hunter 2006].

[1] Typically 1500 m/s in salt water.



(a) Narrow-beam sonar.



(b) Synthetic aperture (wide-beam) sonar.

Figure 1.1: A comparison of the operation of standard narrow-beam side-looking sonar and wide-beam synthetic aperture sonar. With narrow-beam sonar, the target appears in a single ping, while it appears in multiple pings when using SAS. Original images from Mulligan [2007].

1.2 Echo detection

It is commonly assumed that any noise present in the received signal is additive white Gaussian noise (AWGN). In this case, the optimal method of detecting echoes is to process the signal with a matched filter, equivalent to correlating the received signal with the transmitted signal [Crocker 1998]. The time delay of any received echoes are indicated by maxima in the correlation, and the height of the peaks — and hence the ability to distinguish features from background noise — is proportional to the energy of the received signal.

The simplest signal to transmit is a gated sinusoid lasting for the desired ping length τ_p. The corresponding cross-correlation has a triangular envelope of width $2\tau_p$. Assuming that the subsequent processing is naïve and requires there to be no overlap between adjacent targets, the across-track resolution is

$$\delta_r = \frac{c\tau_p}{2}, \tag{1.3}$$

i.e., the resolution is directly proportional to the ping length [Caprais and Guyonic 1997]. Shortening the length of the ping to improve the resolution requires an increase in the instantaneous power to maintain the total energy of the signal. However, the presence of non-linear effects such as cavitation place an upper limit on the power which can be transmitted into water [Urick 1975].

Pulse compression is a technique which achieves a high resolution with a long duration pulse by modulating the frequency or phase of the signal to increase its bandwidth B [Hovanessian 1980]. The cross-correlation of such a signal is proportional to $\operatorname{sinc} B\tau_p$, and has a main lobe of width $1/2B$ [Skolnik 1980]. The use of pulse compression gives a across-track resolution of

$$\delta_r = \frac{c}{2B}, \tag{1.4}$$

i.e., the duration of the pulse has no effect on the resolution [Hayes and Gough 1992].

1.3 KiwiSAS-IV

The current SAS system operated by the Acoustics Research Group at the University of Canterbury is known as KiwiSAS-IV. It transmits the signal using a 3×12 array of lead zirconate titanate (PZT) transducers and receives the echo with a 3×3 array of polyvinylidene fluoride (PVDF) transducers [Hayes 2003]. These transducer arrays are mounted in a neutrally buoyant towfish along with the transmitter and receiver electronics. The received data is stored on a hard disk for post-processing, and a 100 Mbit/s Ethernet link to the tow vessel allows the system configuration to be altered and some of the raw data monitored. The ping transmitted by KiwiSAS-IV consists of two superimposed linear frequency chirps, one varying from 20–40 kHz and the other from 90–110 kHz.

1.4 Research goal

The aim of this thesis is to develop a replacement for the front-end electronics in the KiwiSAS-IV system. The main goal is to reduce the weight, size, and power requirements to a level where the sonar can be mounted on an autonomous underwater vehicle (AUV); such improvements are also beneficial for the existing towed system. An AUV would allow the mapping of large and/or remote areas of the seafloor without the need for human interaction; in deep water a towed system becomes difficult to deploy for such a task. Additionally, an AUV will enable the imaging of more dangerous waters where the current system cannot be used.

The new system will use a single transducer array for both transmitting and receiving. The amplifiers used to generate the transmitted signal will be replaced with smaller and more efficient class D amplifiers, while the highly integrated AD9271 integrated circuit (IC) will implement a large portion of the receiver. A field programmable gate array (FPGA) is required to interface between the back-end system (which will provide the data storage, power supplies and user interface) and the front-end electronics.

1.5 Assumed knowledge

The preceding overview of the operation of a SAS system should be sufficient to understand the design of the new front-end. Further information about SAS, including various image reconstruction techniques, is given in sources such as Hawkins [1996], Callow [2003], Barclay [2006], and Hunter [2006], and the current status of SAS is reviewed in Hayes and Gough [2009].

It is assumed that the reader has some knowledge of signal processing techniques, especially digital filtering, and their application in embedded systems. The Noble identities are used when discussing the multirate systems used in the design; a good overview of these identities can be found in Milić [2009]. An understanding of programming FPGA systems using very high speed integrated circuit hardware description language (VHDL) would also be beneficial.

1.6 Thesis overview

Chapter 1 presents an introduction to SAS and the goal of this project.

Chapter 2 discusses the construction of the transducers used in the SAS system and the characterisation and modelling of their frequency response.

Chapter 3 details the choice of the system-level components (the receiver, decimation filters and the class D amplifiers).

Chapter 4 covers the design of a prototype front-end and presents the results of hardware testing.

Chapter 5 deals with the FPGA implementation of the necessary firmware to support the electronics. The performance of the system is also discussed.

Chapter 6 draws conclusions on the project and suggests directions for further work to develop it into a fully functional SAS system.

CHAPTER 2

TRANSDUCER CHARACTERISATION

Nearly all men can stand adversity, but if you want to test a man's character, give him power.

Abraham Lincoln

CHAPTER 2

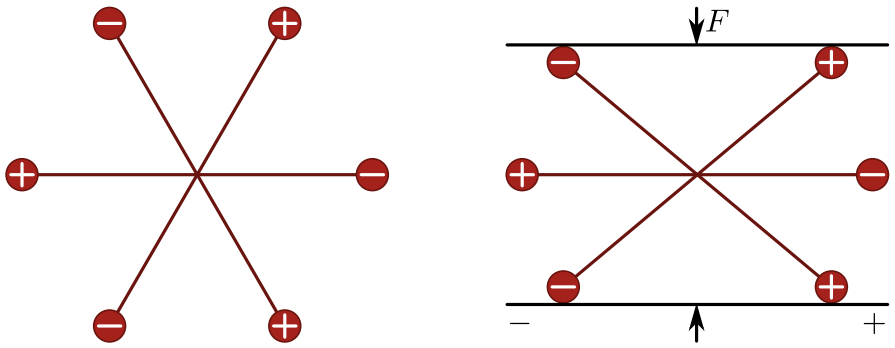
TRANSDUCER CHARACTERISATION

A transducer is a device designed to convert energy from one form into another. For a sonar system, electroacoustic transducers (i.e., those that convert electrical energy to acoustic energy and vice-versa) are usually used. The KiwiSAS-IV transducers utilise the reverse piezoelectric effect to convert electrical energy to mechanical energy. The mechanical energy is then used to displace the radiating face of the transducers to generate the acoustic energy. Receiving the echo signal works in reverse; the acoustic energy displaces the transducer and the resulting mechanical energy is converted to an electric potential difference by the piezoelectric effect.

The piezoelectric ceramic discs used in the KiwiSAS-IV transducers exhibit a number of resonant frequencies. While some of these resonances are used to transmit and receive the desired signal, there are others which, if unintentionally excited, could cause a large power dissipation and damage the transducers or electronics. In order to prevent this, a study of the transducers was undertaken to locate these resonant frequencies. From this information, an electrical model of the transducers was developed.

2.1 The piezoelectric effect

The direct piezoelectric effect is the ability of certain materials to generate an electric potential in response to an applied force. It was discovered by the Curie brothers who demonstrated it on quartz crystals [Curie and Curie 1880]. A simple molecular model of the piezoelectric effect is shown in Figure 2.1. In its unperturbed state (Figure 2.1a), the symmetry of the molecule leads to the dipoles cancelling so there is no overall charge. However, applying an external force (Figure 2.1b) deforms the molecule, with



(a) No external force. (b) External force applied.

Figure 2.1: A simple example of the direct piezoelectric effect in a single molecule. With no external force the symmetry of the molecule leads to the dipoles cancelling so that the molecule has no overall charge. Applying an external force disturbs this symmetry and leads to the creation of an electric field.

the loss of symmetry leading to the creation of an electric field. When replicated across the many molecules of a piezoelectric material, this field causes a build-up of charge on the surface of the material with the magnitude of the charge proportional to the applied force. Hankel [1881] proposed the name piezoelectricity, meaning "electricity by pressure"[1].

The piezoelectric effect is reversible, i.e., an applied electric field can be used to deform the material. This reverse piezoelectric effect (also referred to as the converse or inverse piezoelectric effect) was predicted by Lippmann [1881] using the principle of conservation of charge. The Curie brothers confirmed the existence of the inverse effect in a later experiment [Curie and Curie 1881].

Ceramic materials do not naturally exhibit the piezoelectric effect; however, they can be conditioned to do so. To achieve this, the material is heated and an electric field applied, causing a permanent expansion in the direction of the applied field. After this, applying tension or compression to the material results in an electric potential being generated (the direct piezoelectric effect) while an electric field applied in the same direction as the conditioning field causes the material to be distorted (the reverse effect). Since ceramic materials can be easily moulded into various shapes, they are generally preferred over crystalline materials in underwater transducers [Burdic 1984].

[1]From the Greek *piezo* or *piezen*, meaning to squeeze or press.

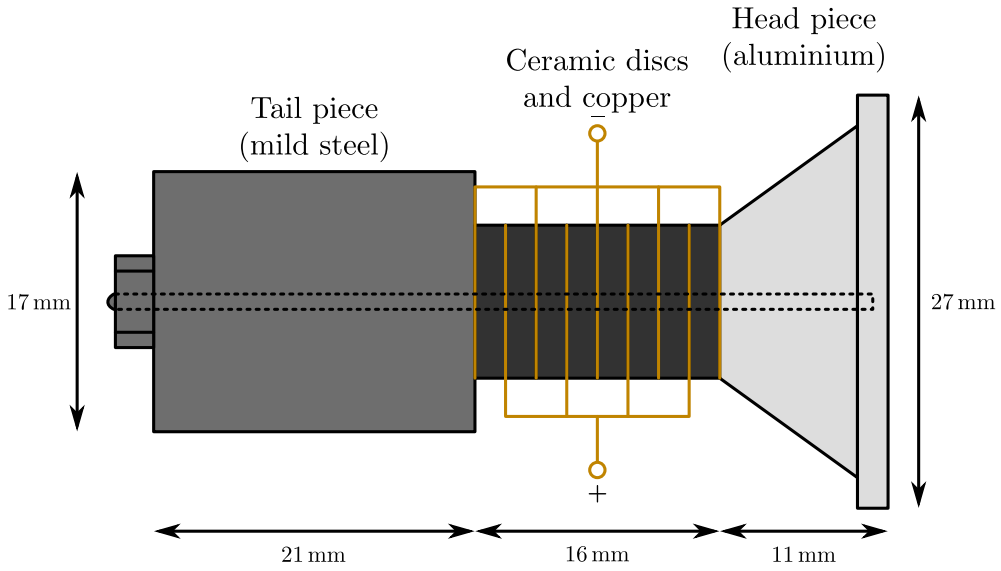


Figure 2.2: A schematic diagram of the Tonpilz transducer design used in the KiwiSAS system. The head piece has a large radiating face and is tapered to meet the ceramic discs. Copper electrodes are placed between the discs to form the electrical port. A bolt runs through the transducer to ensure the ceramic discs are kept under compression. Original image from Knight [1987].

2.2 Construction

One of the fundamental mechanoacoustic oscillator designs is the Tonpilz design, named by Hahnemann and Hecht [1920] from the German words meaning 'sound' and 'mushroom'. The transducers for the KiwiSAS system were designed as Tonpilz transducers by Peter Gough at the University of Canterbury; the design is described in Knight [1987] and Gough and Knight [1989]. They consist of eight acoustically active ceramic discs sandwiched between head and tail pieces. The discs are made from PZT, a piezoelectric ceramic compound of lead, zirconium, titanium, and oxygen, and are each 10 mm in diameter and 2 mm thick. Thin copper electrodes are situated between each disc and alternately soldered to copper bars, thus forming the electrical port of the transducer by connecting the ceramic discs in parallel. The discs are weak in tension so a steel stress rod is placed through the centre of the transducer to ensure they are always under compression; this also ensures a good mechanical contact between the various components. A schematic diagram of the transducer construction is given in Figure 2.2.

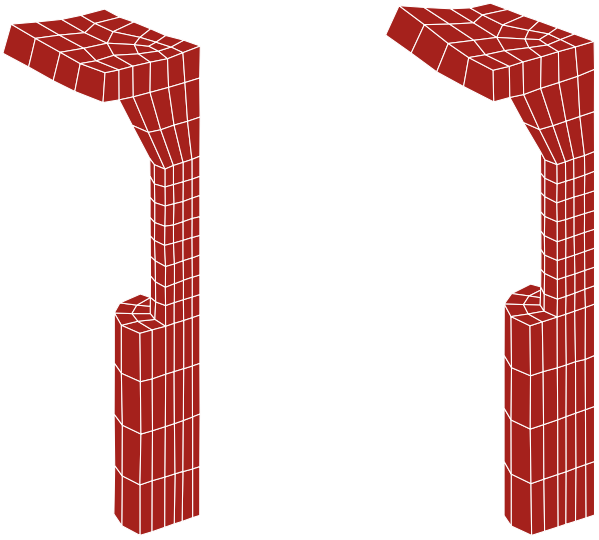
To maximise the acoustic energy transfer between two media, their acoustic impedances

should match. However, PZT has an acoustic impedance around 20 times that of water; to help reduce this mismatch the head piece is constructed from aluminium which has an acoustic impedance between the two. This head piece tapers from the 27 mm square radiating face to the 10 mm diameter of the ceramic discs, with the larger area of the radiating face improving the impedance match with the water (as well as providing the distinctive 'mushroom' shape of a Tonpilz design). By contrast, the tail piece is made from mild steel which has a higher acoustic impedance than PZT. This, coupled with the fact that the tail piece radiates into air, causes a large impedance mismatch in order to reflect energy back to the head piece to ensure that most of the transducer displacement occurs at the radiating face.

2.3 Finite element modelling

When characterising a complex structure, it is often difficult or impossible to do so analytically. In these situations, discrete methods — where the system is broken into a number of well-defined components — are often used. The behaviour of these components are commonly described by differential equations; a number of general techniques for solving these equations have been developed, such as finite difference approximations or weighted residual methods [Zienkiewicz and Taylor 2000]. The finite element method (FEM) is a numerical technique for solving differential equations and is often preferred for discrete modelling over other methods, due to its ability to better handle complicated geometries [Lerch and Kaarmann 1987]. The FEM can be used to model the behaviour of piezoelectric transducers [Allik et al. 1974], and, in conjunction with the boundary element method (BEM), can characterise the behaviour of the transducer under water loading [Lerch et al. 1992].

Hawkins and Gough [1996] used the FEM to model the KiwiSAS-IV transducers. Their results give two resonances, one at 24.1 kHz and the other at 37.7 kHz, which were validated by testing of the constructed transducer. They also used the FEM to predict the displacement of the transducer radiating face at these frequencies. These predictions are shown in Figure 2.3. While this model could be extended to predict the resonances at higher frequencies, the FEM is a complex technique which relies on



(a) 24.1 kHz (b) 37.7 kHz

Figure 2.3: FEM models of the displacement of the transducer's radiating face at two resonant frequencies; due to the symmetrical shape, only a quarter of the transducer needs to be modelled. Original images from Hawkins and Gough [1996].

estimates of the transducer shape and the properties of the construction materials to generate these predictions. As such, it was considered more appropriate to construct a model based around measurements taken from the transducers.

2.4 Butterworth-Van Dyke model

Butterworth [1914] considered general capacitive electromechanical systems and concluded that the electrical admittance of such a system can be represented as a capacitor in parallel with a series RLC network. Van Dyke [1928] showed that this model could be applied to piezoelectric crystals and that it could be extended to represent the various vibration modes of a piezoelectric material as shown in Figure 2.4. This Butterworth-Van Dyke (BVD) model uses a capacitor to model the bulk capacitance of the material in parallel with a number of series RLC branches to model the vibration modes. This assumes that when one branch is resonating the others all have high impedance and thus requires the resonant frequencies to be sufficiently separated. Due to the high Q of most piezoelectric materials this is not usually an issue [Van Dyke 1928]. The main benefits of using the BVD model are that its values can be determined solely

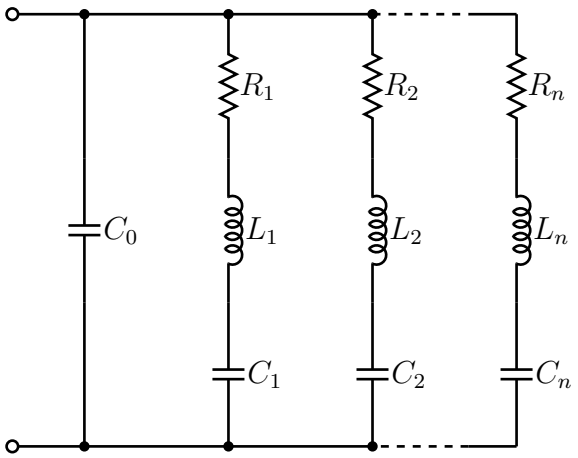


Figure 2.4: The Butterworth-Van Dyke model of a piezoelectric material.

from knowledge of the electrical admittance of the transducer and that it can easily be used in any circuit simulator. However, as this is an electrical model, it will not take mechanical resonances into account. For instance, the mode 3 resonance of the KiwiSAS-IV transducers is purely mechanical [Hawkins and Gough 1996].

More accurate models can be developed by considering the entire electromechanical system and using the electromechanical analogies described by Mason [1948] to form an equivalent electrical circuit. The Mason model was one of the earliest such models, and was later extended by Redwood [1961] to form the Redwood model. One of the most common models is the KLM model, proposed by Krimholtz et al. [1970]. An overview of various piezoelectric transducer models can be found in Arnau [2004]. As the point of this study is to model the electrical port of the transducer, the BVD model is sufficiently accurate.

2.4.1 Admittance measurements

The admittance of a transducer at a particular frequency can be measured with an admittometer. The transducer is placed in series with a known resistance R to form a voltage divider and an input voltage $v_s(t)$ is applied to the network at the frequency of interest. By measuring the amplitude and phase (with respect to the input) of the voltage $v_t(t)$ across the transducer, the admittance of the transducer at this frequency

can be calculated using the formula

$$Y_T = \frac{V_S - V_T}{V_T} R, \tag{2.1}$$

where V_S and V_T are the phasor representations of $v_s(t)$ and $v_t(t)$ respectively.

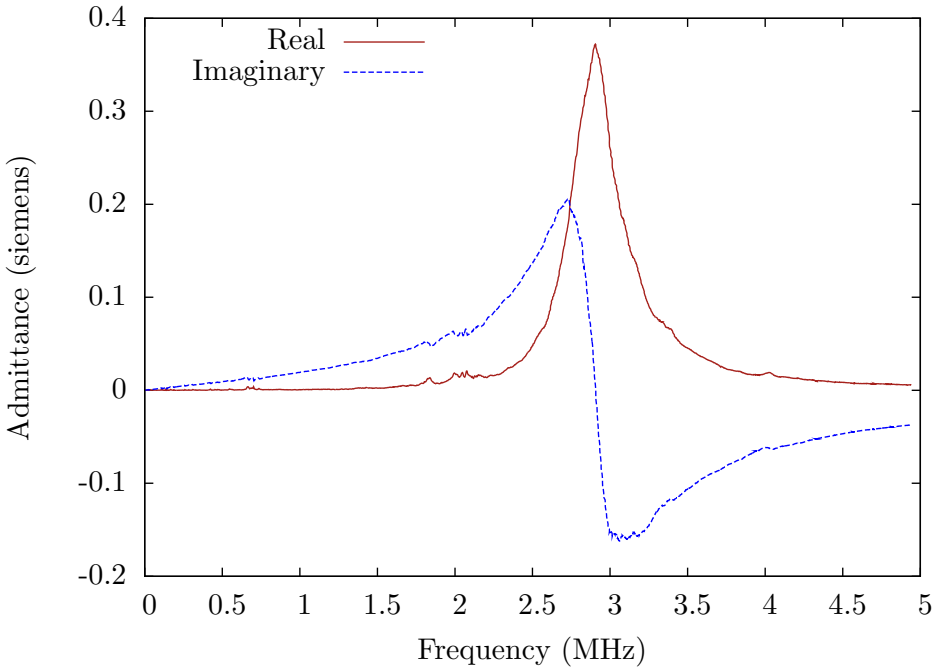
A spare KiwiSAS-IV transducer was placed in a waterproof housing and an admittometer used to measure its admittance from DC to 5 MHz at 500 Hz intervals. These measurements, shown in Figure 2.5a, reveal a large resonance corresponding to the resonant frequency of the piezoelectric crystals at 2.9 MHz as well as smaller resonances around 700 kHz and 2 MHz. The magnitude of the 2.9 MHz resonance means it is important to ensure the electronics produce no significant energy at this frequency. With the impedance being about 2.7 Ω, even a relatively small voltage could dissipate a large amount of power.

A second set of measurements were taken between DC and 200 kHz with a resolution of 20 Hz and are shown in Figure 2.5b. They show the various resonant modes of the transducer, including the 24.1 kHz and 37.7 kHz resonances reported by Hawkins and Gough [1996]. The band between 90 kHz and 110 kHz that is also used for operation contains some resonances, and there are further resonances at still higher frequencies. The smallest impedance in these measurements is approximately 300 Ω at 140 kHz. This, coupled with the fact that there are not likely to be harmonics at these frequencies, means these resonances pose no risk to the system.

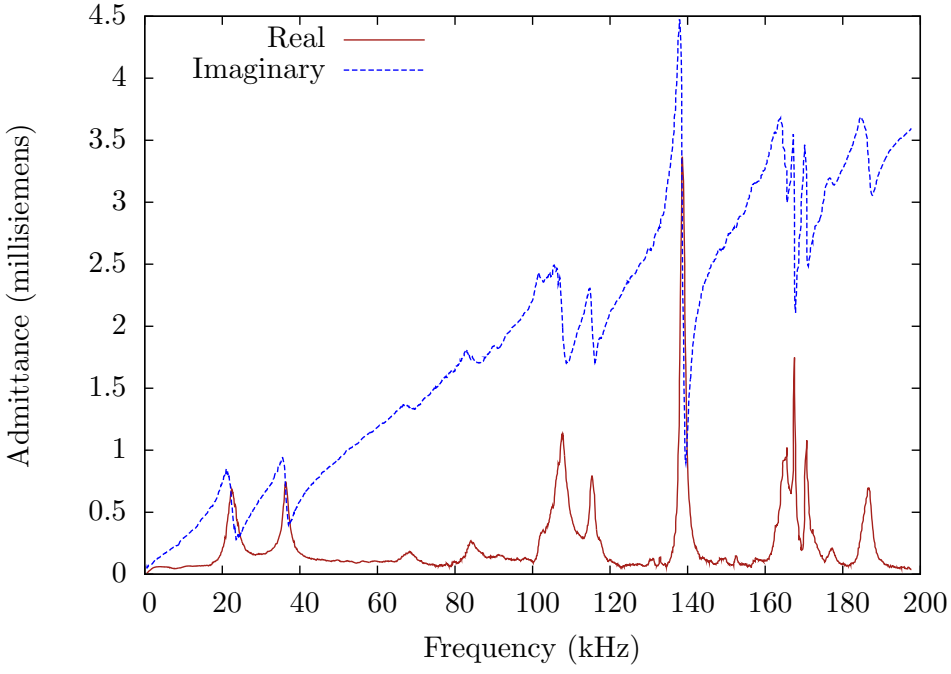
2.4.2 Model fitting

Three parameters — the resistance, inductance, and capacitance – must be found for each branch of the BVD model. Making the assumption that the branches are independent (i.e., that they only have an effect on the overall response around their own resonant frequency and have infinite impedance elsewhere), the resistance can be found from the inverse of the real part of the admittance at resonance. The bandwidth of one of the branches is given by the formula

$$\Delta f_n = \frac{R_n}{2\pi L_n}, \tag{2.2}$$



(a) From DC to 5 MHz



(b) From DC to 200 kHz

Figure 2.5: The admittance of the transducer as a function of frequency.

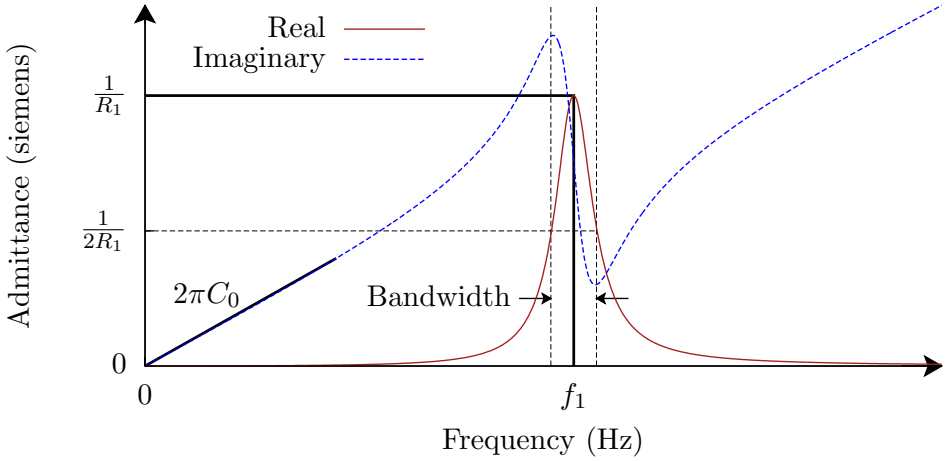


Figure 2.6: Estimating the Butterworth-Van Dyke model parameters from the admittance. From the bandwidth, resonant frequency and real admittance at resonance, the parameters for that branch can be calculated, while the slope of the imaginary admittance at low frequencies provides an estimate for C_0.

where the subscript n refers to the branch in question. As illustrated in Figure 2.6, the bandwidth can be measured from the graph of the admittance, either from where the real part drops to half its value at resonance or from the local maxima and minima in the imaginary part. From this, the value of the inductor for the branch can be calculated using (2.2). The resonant frequency of a branch can be found using the equation

$$f_n = \frac{1}{2\pi\sqrt{L_n C_n}}. \tag{2.3}$$

Hence measuring the resonant frequency allows the capacitance of the branch to be estimated.

The final model parameter to be found is the bulk capacitance C_0. The admittance of an ideal capacitor as a function of frequency is

$$Y_C(f) = j2\pi f C. \tag{2.4}$$

If the imaginary component of this is plotted against frequency, it results in a straight

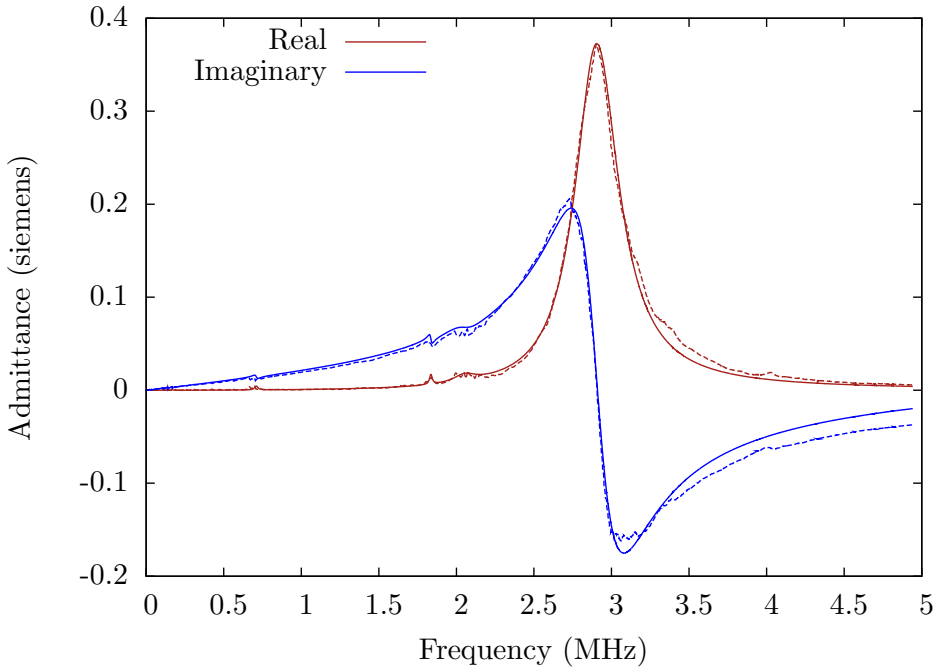
line with the slope and capacitance linked by the relationship

$$C = \frac{m}{2\pi}, \tag{2.5}$$

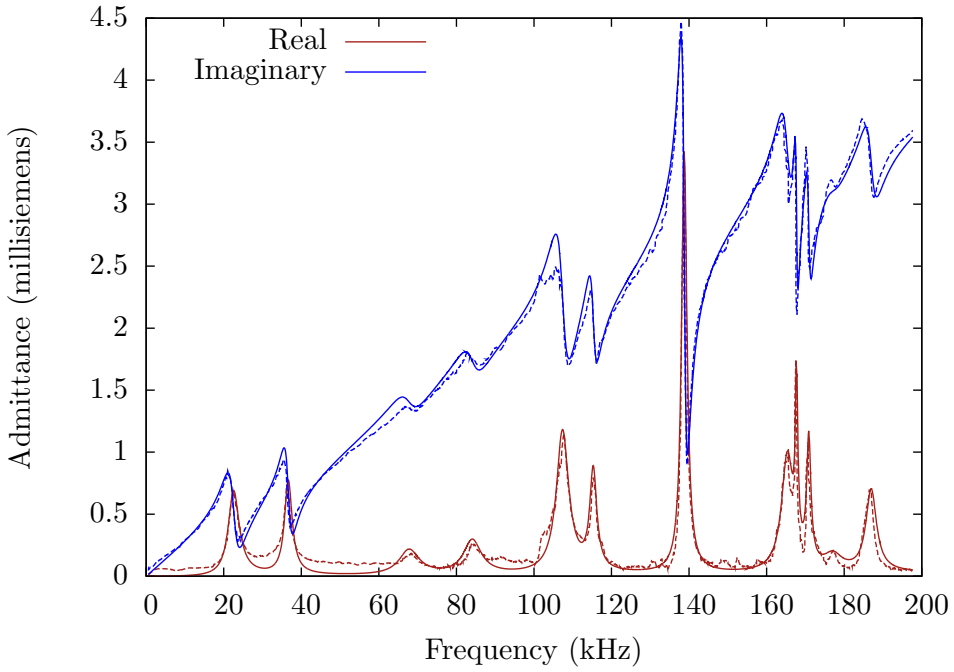
where m is the slope of the line in farads per hertz. Using the same assumption about the independence of the branches as before, the admittance at low frequencies will be dominated by the bulk capacitor. By measuring the slope of the imaginary admittance at low frequencies (i.e., before the resonant branches start to have an effect), the bulk capacitance can be estimated using (2.5).

The values estimated by this method will generally require adjusting, mainly to compensate for the fact that the resonant branches are not independent. This adjustment can be done by hand or by using a software optimisation routine to minimise the difference between the measured admittance and the model admittance to within a suitable degree of tolerance.

A BVD model was generated to fit the measured admittance of Figure 2.5. The initial values were found using a MATLAB script which used a thresholded peak finding algorithm to locate the resonant frequencies in the measured data. The measured bandwidth and admittance at these frequencies was then used to generate the branch values. This resulted in seventeen branches being created to model the various resonances over the whole frequency range. The slope of the imaginary admittance was measured using a best fit technique to generate an estimate of the bulk capacitance. The values returned by this script were then adjusted by hand to improve the match. The resulting values are shown in Table 2.1 and the admittance of the model is compared to the measured admittance in Figure 2.7. This shows a good match in the real admittances at the resonant frequencies with a small error between resonances. The imaginary admittances do not match quite so well. However, the model parameters were unable to be adjusted to give a better match than this.



(a) From DC to 5 MHz



(b) From DC to 200 kHz

Figure 2.7: The admittance of the Butterworth-Van Dyke model of the transducer (solid lines) compared to the measured values (dashed lines).

n	R_n	L_n	C_n	n	R_n	L_n	C_n
0	—	—	630 pF	1	1.46k Ω	67.73 mH	734.00 pF
2	1.32 kΩ	88.04 mH	214.08 pF	3	4.92 kΩ	123.98 mH	44.43 pF
4	3.62 kΩ	103.98 mH	34.43 pF	5	868.9 Ω	35.46 mH	61.96 pF
6	1.24 kΩ	90.85 mH	20.98 pF	7	294.0 Ω	27.75 mH	47.40 pF
8	1.08 kΩ	7.20 mH	19.63 pF	9	766.7 Ω	155.49 mH	5.79 pF
10	993.8 Ω	122.22 mH	7.10 pF	11	7.96 kΩ	23.20 mH	3.62 pF
12	1.51 kΩ	73.51 mH	9.86 pF	13	247.3 Ω	1.51 mH	33.34 pF
14	95.7 Ω	717.60 μH	10.48 pF	15	122.2 Ω	112.19 μH	53.57 pF
16	2.68 Ω	1.25 μH	2.40 nF	17	98.4 Ω	110.73 μH	14.17 pF

Table 2.1: The values of the BVD model generated for the transducer.

CHAPTER 3

SYSTEM DESIGN

What I cannot create, I do not understand.

Richard Feynman

CHAPTER 3

SYSTEM DESIGN

This chapter presents the top-level design of the new front-end system, a block diagram of which is shown in Figure 3.1. The back-end system interfaces with an FPGA, allowing it to control the transmission and receive the returned signal. Class D amplifiers are used to generate the 200 V_{p-p} signals which drive the 72 channels on the two transducer arrays. A bank of transmit/receive (T/R) switches prevent the transmitted signal damaging the receiver electronics, which are based around the AD9271 amplifier and analogue to digital converter (ADC) chip. The sampled data is passed back to the FPGA which decimates it before transferring it to the back-end system for storage. The following sections provide a background to the various blocks in the system and detail their design.

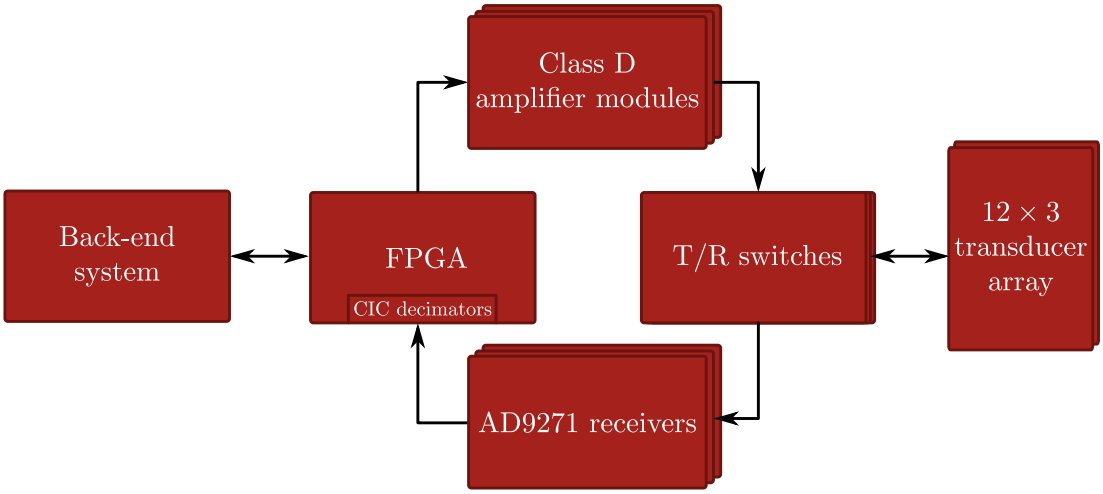


Figure 3.1: A block diagram of the proposed system.

3.1 AD9271 receiver

With the amplitude of the echo signal typically being on the order of millivolts, the first requirement for the receiver is to use a low noise amplifier (LNA) to boost this voltage without introducing significant noise. To this end, each channel in the KiwiSAS-IV receiver uses an INA111 instrumentation amplifier — chosen because of its low noise current — to increase the signal by a factor of 40 dB. Due to the distance of these amplifiers from the ADCs, a pair of THS4140 differential amplifiers with no gain are also employed in each channel to maintain the signal-to-noise ratio (SNR). Each signal is then sampled at 312.5 kHz with a 16-bit resolution by an AD7723 delta-sigma (Δ-Σ) ADC.

This arrangement requires four ICs and the associated discrete components for each channel. In order to reduce the space requirements, the new design utilises the Analog Devices AD9271, a receiver IC primarily designed for medical ultrasound applications. This chip provides eight channels, each with a LNA and a variable gain amplifier (VGA) offering a total gain path that can be varied between 8.4 dB and 42.4 dB. Each channel is then simultaneously sampled by a 12-bit pipelined ADC at a rate which can be adjusted between 10 MHz and 50 MHz. The sampled data is serialised and provided on eight low-voltage differential signalling (LVDS) outputs, one for each channel. A data clock and a frame synchronisation clock are also provided on LVDS links to allow the data to be read. In order to halve the required clock speed, the data is output using double data rate (DDR) transfers, i.e., on both the rising and falling edges of the clock.[1] A block diagram of the AD9271 is shown in Figure 3.2.

There are two downsides to using the AD9271. Firstly, the high sampling rate generates a large amount of data. Even at the slowest sampling rate of 10 MHz, 72 channels sampled with 12 bit resolution equates to a total data rate of 8.64 Gbit/s. Since the highest frequency of interest in the echo signal is 110 kHz, the sampling rate — and therefore the data rate — is higher than needed to represent the signal. The second downside is that the resolution is lower than in the KiwiSAS-IV system. Both

[1] 'AD9271 Octal LNA/VGA/AAF/ADC and Crosspoint Switch Data Sheet (Rev. B)', Analog Devices, May 2009. `http://www.analog.com/static/imported-files/data_sheets/AD9271.pdf`

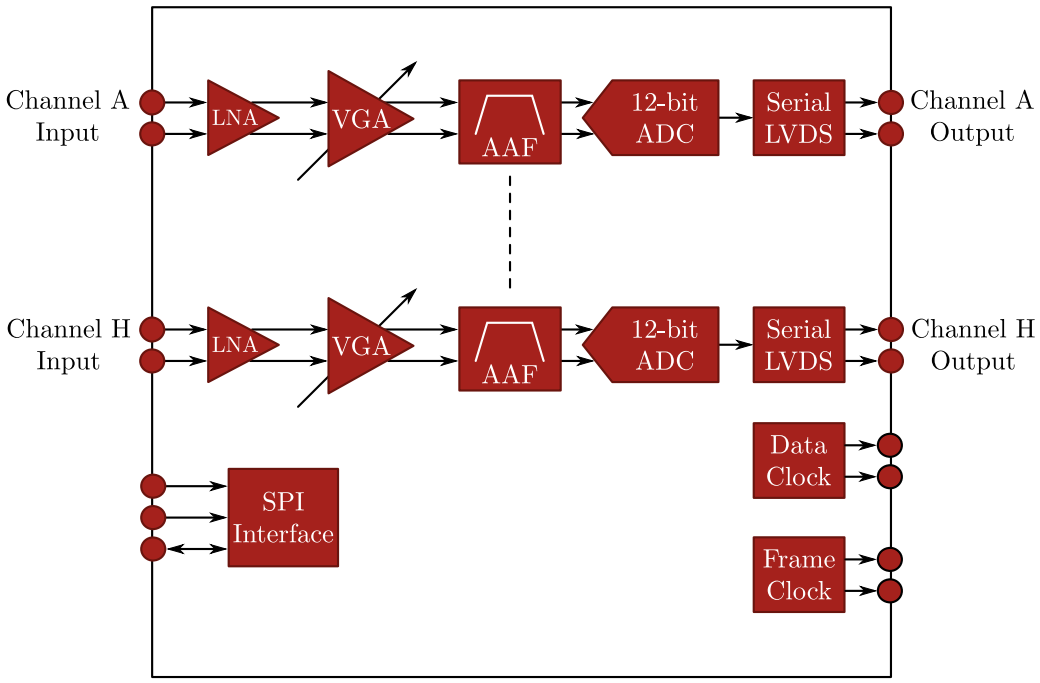


Figure 3.2: A block diagram of the AD9271 IC. Not shown are the inputs to set the gain of the VGAs, the continuous wave Doppler outputs (unused in this design), or the various reference and power supply inputs.

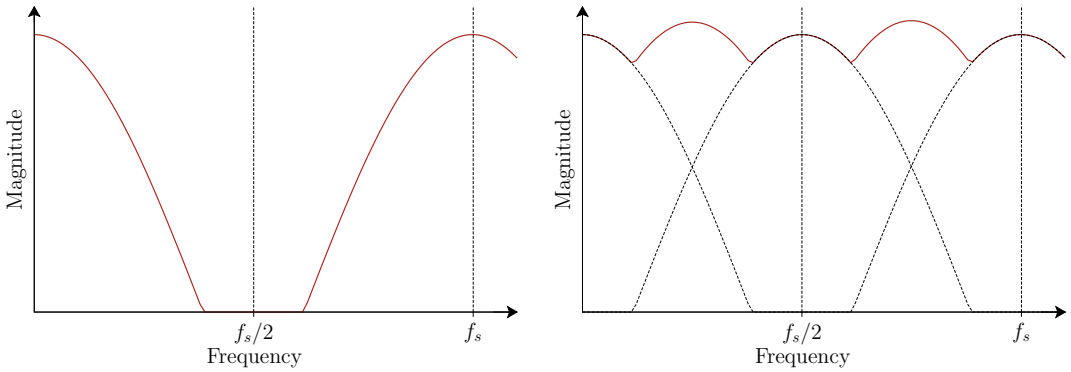
of these problems can be overcome by decimating the signal after it has been sampled.

3.2 Hardware-efficient decimation

In digital signal processing, decimation is the process of reducing the sampling rate of a signal.[2] Lowering the sampling rate reduces the speed that subsequent processing sections must run at, with the minimum possible sampling rate for a system given by the Nyquist limit of the signal. Decimation can also be used to increase the resolution of the sampled data. This is the principle behind a Δ-Σ ADC which takes a single- or dual-bit data stream at a relatively high sample rate and converts it to a multi-bit set of samples at a lower rate.

To reduce the sampling rate by an integer factor R, one out of every R samples is kept and the rest discarded; this process is known as downsampling or subsampling. However, this lowers the Nyquist limit by R and can cause aliasing in the signal.

[2]Strictly speaking, decimation means a 10% reduction in the sampling rate but it is commonly used to refer to any reduction.



(a) Original signal.

(b) Aliased signal after downsampling.

Figure 3.3: An example of aliasing being caused by downsampling. When the original sampling rate is halved, the spectrum overlaps and adds, creating distortion in the signal.

Aliasing is where energy above the new Nyquist limit is folded back into the signal causing distortion; this is illustrated in Figure 3.3. To prevent aliasing, a low-pass filter must be used prior to the downsampling to remove any energy above the new Nyquist limit.

Since this anti-aliasing filter must operate at the higher sampling rate, it can be computationally expensive to implement. For this reason special filter architectures have been developed to reduce this cost. The following sections provide an overview of some of these filters, concluding with the cascaded integrator-comb filter chosen for this design, and compares their performance to a standard low-pass finite impulse response (FIR) filter.

3.2.1 Interpolation

Interpolation, in which the sampling rate of a signal is increased by an integer amount L, is the complementary operation to decimation. It is also a two-step process; first the sampling rate is increased by adding L zeroes between each sample in a step usually referred to as upsampling. This causes images to appear in the spectrum of the faster signal which must then be removed by a low-pass filter to complete the interpolation. Since this anti-image filter has the same desired frequency response as an anti-aliasing filter, and will also be operating at the high sampling rate of the system, the filters

discussed subsequently can also be used for interpolation, usually with little or no modification.

Both decimation and interpolation are only able to alter the sampling rate by integer factors. To adjust the sampling rate by a rational fraction L/R, an interpolation by L is followed by a decimation by R; if performed in reverse, data is lost in the decimation prior to the interpolation. Additionally, performing them in this order allows the anti-aliasing and anti-image filters to be combined so that the overall system is an upsampler followed by a filter and then a downsampler.

3.2.2 Resolution increase

A low-pass filter can be viewed as averaging the input data to form the output. This allows the resolution of the output data to be greater than the input. For example, if each pair of data in the set

$$x[n] = \{3, 4, 4, 6, 6, 5, 5, 3\}$$

is averaged, the resulting output is

$$y[n] = \{3.5, 5.0, 5.5, 4.0\}\,.$$

Averaging each pair of this set then gives

$$z[n] = \{4.25, 4.75\}\,,$$

i.e., each time the number of samples is halved an extra significant digit of resolution is available. This works similarly with a binary system; each time the sampling rate is halved with averaging, an extra bit of resolution is available. This means that the number of extra bits created from decimating by a factor R is

$$B_G = \lceil \log_2 R \rceil\,, \tag{3.1}$$

where $\lceil a \rceil$ is the smallest integer not less than a. Although in theory the resolution can be increased indefinitely by continuing to reduce the sampling rate, in reality the amount of noise present in the system limits this. Beyond a certain point the extra bits just contain noise rather than useful information. The effective number of bits (ENOB) that can be extracted from a sampled signal depends on the signal-to-noise-and-distortion ratio (SINAD). For an ADC performing uniform quantisation, the ENOB is given by

$$\text{ENOB} = \frac{\text{SINAD} - 1.76\,\text{dB}}{6.02}, \tag{3.2}$$

assuming that the error introduced by the quantisation is uniformly distributed [Kester 2005].

3.2.3 Notation

When discussing and analysing a system where the sampling rate changes within the system, it is important to distinguish the different rates. Throughout this thesis f_s refers to the high sampling rate. If the rate change factor R denotes the integer factor by which the sampling rate is decreased, the low sampling rate is then given by fs/R. When discussing filter design, the edge of the passband is denoted by f_{pass} and the edge of the stopband is denoted by f_{stop}. The transition width of the filter is then given by

$$\Delta f = f_{\text{stop}} - f_{\text{pass}}. \tag{3.3}$$

The desired stopband attenuation is denoted by A and specified in decibels.

3.2.4 Anti-aliasing filter design

The SNR of the signal entering the ADC of the AD9271 is dependent on the transmitted signal strength, the distance to the target that caused the echo, the background ocean noise, and the electronic noise of the front-end electronics prior to the ADC. Once in a digital format, the maximum possible SNR is determined by the number of bits N

Parameter	Value
Input sampling rate	10 MHz
Output sampling rate	625 kHz
Sampling rate reduction factor	16
Input resolution	12 bits
Output resolution	16 bits
Passband edge	110 kHz
Stopband attenuation	100 dB

Table 3.1: The desired parameters for the anti-aliasing filter.

used to represent the signal, and is given by

$$\text{SNR} = 6.02N + 1.76\,\text{dB}, \tag{3.4}$$

assuming that any error in the quantisation is uniformly distributed [Kester 2005]. If the signal is to be decimated to give a 16-bit resolution, (3.4) gives the maximum SNR as 98.08 dB. To avoid adding any noise above this level, the anti-aliasing filter should have a stopband attenuation A of at least 100 dB.

To increase the resolution by 4 bits, (3.1) shows that the sampling rate must be reduced by a factor of 16. If the AD9271 is used at its minimum sampling rate of 10 MHz, this corresponds to a reduced sampling rate of 650 kHz. The passband edge of the required anti-aliasing filter is 110 kHz, the highest frequency of interest in the echo signal. The desired filter parameters are summarised in Table 3.1.

3.2.5 FIR filters

The simplest — and most inefficient — way to implement the anti-aliasing filter is with a standard FIR filter. The n^{th} output of such a filter is given by

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k], \tag{3.5}$$

where N is the number of taps in the filter and $h[k]$ is the coefficient of the k^{th} tap in the filter. The number of taps required to implement a filter depends on its desired

response. A simple formula to estimate N, given in Harris [2004], is

$$N = \frac{f_s}{\Delta f}\frac{A}{22}. \tag{3.6}$$

There are many methods of obtaining the tap coefficients for a filter. Common ones include the frequency sampling method [Rabiner and Schafer 1971], the optimal method introduced by Parks and McClellan [1972], or by applying a windowing function to the ideal response [Rabiner and Gold 1975].

If implemented as a FIR filter, the stopband edge of the anti-aliasing filter would be 312.5 kHz, corresponding to a transition band of 202.5 kHz. For a stopband attenuation of 100 dB, (3.6) gives the number of taps required as 225. An inspection of (3.5) shows that $N-1$ additions and N multiplications are required to generate each output. Hence using FIR anti-aliasing filters would require approximately 1.61×10^{11} additions and 1.62×10^{11} multiplications per second to decimate all 2×36 channels.

3.2.6 Symmetric FIR filters

If the i^{th} and j^{th} tap coefficients are the same, then (3.5) can be expanded and these terms collected to give

$$y[n] = h[0]x[n] + \cdots + h[i]\Big(x[n-i] + x[n-j]\Big) + \cdots . \tag{3.7}$$

Although this requires the same number of additions as before, the number of multiplications has been reduced by one.

The ideal low-pass filter passes unmodified all frequencies below f_{pass} and rejects all frequencies above it, i.e., the frequency response is a rectangular pulse. The corresponding impulse response — found by taking the inverse Fourier transform of the frequency response — is a sinc function. Since the impulse response is equivalent to the tap coefficients, this means the tap coefficients are symmetric around the centre coefficient. Although this ideal frequency response is not realisable, most low-pass filters have tapered rectangular frequency responses which also correspond to a symmetric impulse response.

Taking advantage of this fact allows the output equation to be rewritten along the lines of (3.7). Hence a symmetric FIR filter with N taps still requires $N-1$ additions per output sample, but the number of multiplications required is reduced to $\frac{N}{2}$ if N is even or $\frac{N+1}{2}$ if N is odd. If the FIR filter from the previous section is designed so that is has symmetric tap coefficients, then the computational effort would be reduced to 1.61×10^{11} additions and 8.14×10^{10} multiplications per second.

3.2.7 Multistage FIR filters

Observing the formula given in (3.6), and noting that A is typically constant for a given system, to reduce N either the sampling rate must be decreased or the transition width increased. One technique for doing this is to split the decimation into several stages. The first stage then operates at the full rate but with a larger transition width, and the following stages have decreased sampling rates, albeit with smaller transition widths. Since they are operating at lower speeds, the number of computations per second in the latter stages is further reduced.

If there are K stages, and the k^{th} stage reduces the sampling rate by an integer factor R_k, then the relationship

$$\prod_{k=1}^{K} R_k = R \tag{3.8}$$

must be satisfied to achieve the overall sampling rate reduction. The first step is selecting the order of the sections and, in some cases, selecting which combination of ratios to use (for example, to achieve an overall reduction of 16, a two-stage implementation could use stages of 2 and 8, 4 and 4, or 8 and 2). Crochiere and Rabiner [1975] formulated an optimisation problem based on K, R and the filter parameters and showed that this resulted in a $(K-1)$-dimensional problem. Coffey [2003] extended this work and showed that the problem could be simplified such that only one-dimensional root finding was required.

For a two-stage system with a total decimation factor of 16, having both stages decimate by a factor of four is the most efficient arrangement. The first stage operates at a sampling rate of 10 MHz, but the stopband edge is at 1.25 MHz for a transition width of 1.14 MHz; applying (3.6) gives $N_1 = 40$. The second stage operates at a

sampling rate of 2.5 MHz with a transition width of 202.5 kHz and requires $N_2 = 57$ taps. Taking into account that the taps of the second stage are operating at a slower rate, implementing these as standard FIR filters requires 3.82×10^{10} additions and 3.91×10^{10} multiplications per second to decimate all 72 channels. Implementing them as symmetric FIR filters lowers the multiplications to 1.96×10^{10} per second. This corresponds to around 24% of the computations required for a single stage decimation.

3.2.8 Half- and M^{th}-band filters

Introduced by Bellanger et al. [1974], a half-band filter is a symmetric, zero-phase FIR filter with an impulse response satisfying the condition

$$h[2n] = \begin{cases} c & n = 0 \\ 0 & n \neq 0 \end{cases}, \tag{3.9}$$

where c is usually chosen to be 0.5. Applying this condition to the transfer function of the filter and converting it into the frequency domain leads to the relationship

$$H(e^{j\omega}) + H(e^{j(\pi-\omega)}) = 2c, \tag{3.10}$$

where ω is the angular frequency, provided that the non-zero coefficients of the filter are real [Vaidyanathan 1990]. From this, it follows that $H(e^{j(\pi/2-\theta)})$ and $H(e^{j(\pi/2+\theta)})$ add up to $2c$ for all values of θ. In other words, the frequency response has symmetry around the half-band frequency $\pi/2$, i.e., it is a lowpass filter with the passband edge at a quarter of the sampling frequency, hence the name half-band filter. This property, along with half the filter coefficients being zero, makes the half-band filter suitable for decimating a signal by a factor of two. Larger decimation ratios can be achieved by chaining multiple half-band stages together. For instance, implementing the anti-aliasing filter for a factor of sixteen decimation requires four cascaded half-band filters with the number of taps being $N_1 = 20$, $N_2 = 20$, $N_3 = 23$ and $N_4 = 29$. Taking advantage of the zero coefficients and symmetry means that, for all 72 channels, 1.30×10^{10} additions and 7.20×10^9 multiplications per second are required, or approximately

8% of the computations for a symmetric FIR filter.

The M^{th}-band filter is a generalisation of the half-band filter. It is defined as a zero-phase symmetric FIR filter with an impulse response satisfying the condition

$$h[Mn] = \begin{cases} c & n = 0 \\ 0 & n \neq 0 \end{cases}, \tag{3.11}$$

where the constant c usually chosen to be $1/M$. This condition is also called the Nyquist property [Mueller 1973]. This leads to the M^{th}-band filter being a lowpass filter with the passband edge being at π/M, making it suitable for a decimation by a factor of M. Due to the zero coefficients, an M^{th}-band filter exhibits a $1/M$ reduction in computational effort when compared to a general FIR filter of the same length [Mintzer 1982]. For instance, using a 16^{th}-band filter as the anti-aliasing filter requires 1.52×10^{11} additions and 7.63×10^{10} multiplications per second, around 94% of the computations needed for a symmetric FIR filter. In general, larger values of M lead to the same problem as a standard FIR filter, namely a high sampling rate coupled with a small transition width, thus requiring a large number of taps to meet the desired specification. Since an M^{th}-band structure only offers a computational saving of $1/M$, it is generally more efficient to use a multistage half-band design.

3.2.9 Polyphase filters

Implementing a filter followed by a decimator is inefficient since only one out of every R outputs from the filter are left after the decimation, with the rest being discarded. The polyphase decomposition, which has its origins in work by Bellanger et al. [1976], is used in many multirate signal applications, and can be applied in this situation to reduce the workload of the filter by only calculating the used outputs. The transfer response of a general digital filter is

$$H(z) = \sum_{k=-N}^{N} h[k] z^{-k}. \tag{3.12}$$

This can be expanded and grouped into odd and even components:

$$\begin{aligned} H(z) = &\left[\cdots + h(-2)z^2 + h(0) + h(2)z^{-2} + \cdots\right] \\ &+ z^{-1}\left[\cdots + h(-1)z^2 + h(1) + h(3)z^{-2} + \cdots\right]. \end{aligned} \tag{3.13}$$

The tap weights can then be split into two groups, one with the even coefficients and one with the odd coefficients. This effectively creates two sub-filters with the transfer functions

$$H_0(z) = \sum_{k=-N/2}^{N/2} h[2k]z^{-k} \tag{3.14}$$

and

$$H_1(z) = \sum_{k=-N/2}^{N/2} h[2k+1]z^{-k}. \tag{3.15}$$

The transfer function of the overall filter can then be rewritten in terms of these sub-filters to give

$$H(z) = H_0(z^2) + z^{-1}H_1(z^2). \tag{3.16}$$

This representation, known as the two-component polyphase decomposition of $H(z)$, is valid for both FIR and infinite impulse response (IIR) filters [Vaidyanathan 1990]. This can then be extended to give the M-component polyphase decomposition of $H(z)$:

$$H(z) = \sum_{m=0}^{M-1} z^{-m}H_m(z^M). \tag{3.17}$$

This splits $H(z)$ into M sub-filters $H_0(z)$, $H_1(z)$, ..., $H_{M-1}(z)$ with the tap weights of the m^{th} subfilter given by

$$h_m[k] = h[kM + m] \tag{3.18}$$

Taking the M-component polyphase decomposition of a decimator gives the system shown in Figure 3.4a. By itself, this does not offer any improvement as each of the sub-filters is still operating at the original sampling rate and the overall number of taps is the same as the original filter. However, by applying the third Noble identity to the polyphase filter, the sampling rate reduction can be moved prior to the filtering as shown in Figure 3.4b. This reduces the number of computations by a factor of M since

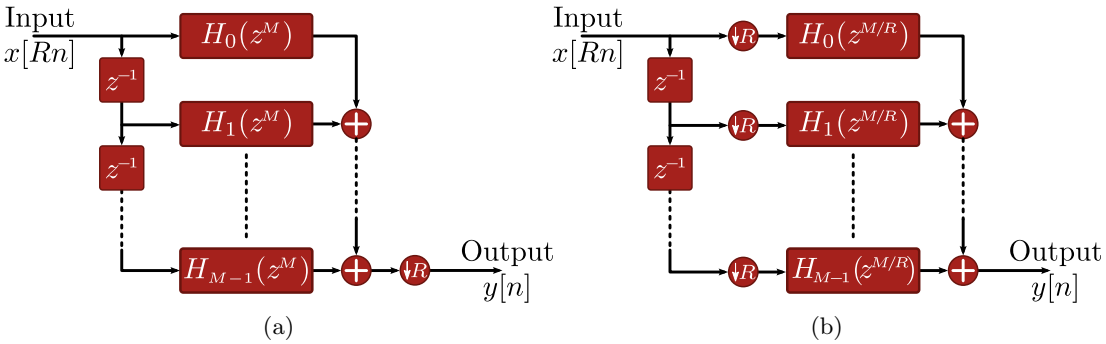


Figure 3.4: Using the polyphase decomposition in a decimator. Applying an M-component decomposition of the filter is shown in (a). To reduce the sampling rate, the third Noble identity is applied to obtain the structure shown in (b).

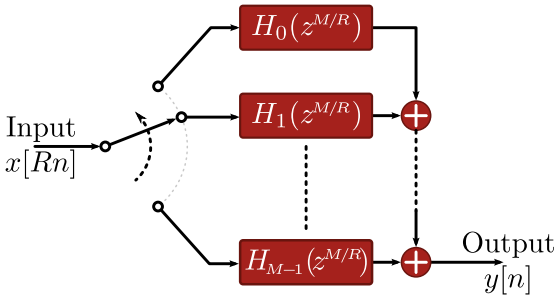


Figure 3.5: The commutator switch implementation of a polyphase filter for decimation.

all the sub-filters now operate at a slower sampling rate.

A common implementation of the polyphase decomposition in a decimator is the commutator switch system introduced by Crochiere and Rabiner [1981] and shown in Figure 3.5. The FIR filter is designed such that the number of taps N is a multiple of R, and then R-component decomposed to give R sub-filters, each with N/R taps. After applying the third Noble identity, the delay line and decimator blocks are replaced with a rotating commutator. In this system the sub-filters are all operating at the same frequency but with a different phase as the inputs are given sequentially rather than concurrently. Providing the overall output is read at the correct time this gives the same results as previously.

Since the sub-filters all operate at the low sampling rate of the system, the number of computations per second are lowered although the number of taps remains the same. It is worth noting that $M - 1$ extra additions are required to combine the outputs

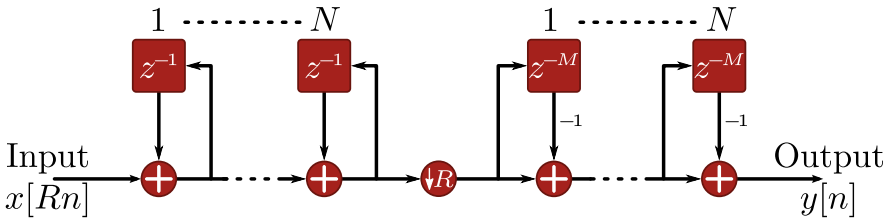


Figure 3.6: The structure of a CIC decimation filter.

of the sub-filters into the final output. To apply the polyphase decomposition to the previously designed FIR filter, the number of taps would first be increased from 225 to 240. A 16-component polyphase decomposition then splits the original filter into 16 sub-filters, each with 15 taps. Due to the splitting of the coefficients, the sub-filters cannot be implemented as symmetric FIR filters.[3] Implementing them as standard FIR filters and accounting for the extra additions results in a total of 1.14×10^{10} additions and 1.08×10^{10} multiplications per second for 72 channels. This corresponds to around 7% of the computations required if a standard FIR filter is used for each channel.

3.2.10 CIC filters

The cascaded integrator-comb filter was first introduced by Hogenauer [1981]. Much as the name suggests, they consist of a block of N integrators cascaded with a block of N comb filters with the change in sampling rate occuring between these two blocks. The integrators are always operated at the high sampling rate and the comb filters at the low sampling rate. For decimation, this means the integrators come first as illustrated in Figure 3.6; for an interpolation filter the comb filters are placed first. As neither the integrators or comb filters require multiplications and only N additions per sample are needed in each of the blocks, a cascaded integrator-comb (CIC) filter requires very little computational effort.

[3] The only case where some symmetry exists is if M is odd; in this situation, the 'middle' sub-filter — e.g., the second sub-filter if $M = 3$ or the third sub-filter if $M = 5$ — will be symmetric.

3.2.10.1 Transfer function

The transfer function of a single integrator is

$$H_I(z) = \frac{1}{1 - z^{-1}}, \tag{3.19}$$

and for a single feed-forward comb filter with delay M the transfer function is

$$H_C(z) = 1 + z^{-M}. \tag{3.20}$$

Since the integrators and comb filters are running at a different speed, they cannot be directly combined to form a transfer function for the complete CIC filter. To do this, the comb filter — which always operates at the slow sampling rate — must be referenced to the high sampling rate. If the CIC decimator is being considered then the third Noble identity is used while analysing the CIC interpolator requires the use of the fifth Noble identity. In either case, the transfer function of the comb filter referenced to the high sampling rate is

$$H_C(z) = 1 + z^{-RM}. \tag{3.21}$$

Since a CIC filter consists of N cascaded integrators and comb filters, (3.19) and (3.21) can be combined to give the transfer function of a CIC filter referenced to the high sampling rate as

$$\begin{aligned} H(z) &= \left[H_I(z)H_C(z)\right]^N, \\ &= \left[\frac{1 - z^{-RM}}{1 - z^{-1}}\right]^N. \end{aligned} \tag{3.22}$$

This can be rewritten as the sum of a geometric series to give

$$H(z) = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N. \tag{3.23}$$

From this, it can be seen that the CIC filter is equivalent to a cascade of N identical FIR filters, each with RM unitary tap coefficients.

3.2.10.2 Stability

To evaluate the stability of the CIC filter, the poles and zeroes of the transfer function given in (3.22) must be found. There is a pole at $z = 1$ repeated N times, i.e., each additional stage in the filter adds another pole at this point. Since this pole lies on the unit circle at the point corresponding to DC in the frequency domain, it can be seen that the pole is the result of the unity feedback in the integrator section of the filter. The locations of the zeroes in the filter are given by

$$\begin{aligned} 1 - z^{-RM} &= 0, \\ z^{RM} &= 1. \end{aligned} \tag{3.24}$$

The complex quantity z can be expressed in the polar form

$$z = re^{j\varphi}, \tag{3.25}$$

where r is the magnitude of z and φ is its phase in radians. Substituting this into (3.24) and taking the magnitude of both sides gives $r = 1$, i.e., the zeroes lie on the unit circle. The phases are found from

$$\begin{aligned} e^{j\varphi RM} &= 1, \\ \cos(\varphi RM) + j\sin(\varphi RM) &= 1, \end{aligned} \tag{3.26}$$

and thus

$$\varphi = \frac{k2\pi}{RM} \qquad k = 0, 1, \ldots, (RM - 1). \tag{3.27}$$

In other words, there are RM zeroes evenly spaced around the unit circle. As with the pole, each of the zeroes is repeated N times, once for each stage in the filter.

For a system to have bounded-input, bounded-output (BIBO) stability, all poles not cancelled by co-located zeroes must lie inside the unit circle. From (3.27), the $k = 0$ pole has a phase $\varphi = 0$ and is therefore located at $z = 1$. Since both this zero and the pole are repeated N times, they cancel each other meaning the CIC filter is BIBO stable.

3.2.10.3 Register growth

When used as an interpolator, the comb filters precede the integrators and pre-condition the data so that the integrators do not overflow [Hogenauer 1981]. However, when used as a decimator, the integrator section of the filter will overflow due to the unity feedback. Since the comb filters calculate the difference between two samples, the overall output will be bounded, provided that a numbering system which wraps between the most positive and most negative numbers (such as two's complement) is used. The numbering system must also be able to handle the maximum output of the comb filters.

The maximum register growth $G_{\max}$ is defined as the magnitude of the largest possible output to the largest possible input. By rewriting the transfer function as a fully expanded polynomial in z^{-1}, Hogenauer [1981] shows that the maximum register growth for a CIC decimator is

$$G_{\max} = (RM)^N . \tag{3.28}$$

In order to be able to accomodate this increase in signal magnitude at the output, the number of bits used in the integrators and combs must be

$$B_G = \lceil N \log_2 (RM) \rceil \tag{3.29}$$

bits greater than the input data.

3.2.10.4 Frequency response

The discrete-time Fourier transform (DTFT) $X(\omega)$ of a discrete signal $x[n]$ is defined as [Ambardar 1999]

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-jn\omega}, \tag{3.30}$$

where ω is the angular frequency. The z-transform $X(z)$ of the same discrete signal is defined as [Ambardar 1999]

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}. \tag{3.31}$$

Using the polar definition of z given in (3.25), this can be written as

$$X(e^{j\varphi}) = r \sum_{n=-\infty}^{\infty} x[n]e^{-jn\varphi}. \tag{3.32}$$

Comparing (3.30) and (3.32) shows that the DTFT is a special case of the z-transform with $r = 1$ and $\varphi = \omega$. In other words, the DTFT can be found by evaluating the z-transform around the unit circle with the substitution

$$z = e^{j\omega}. \tag{3.33}$$

Using this substitution on the CIC transfer function of (3.22) gives

$$H(\omega) = \left(\frac{1 - e^{-jRM\omega}}{1 - e^{-j\omega}}\right)^N. \tag{3.34}$$

Both the numerator and denominator are in the form $1-e^{-j\alpha}$ which can be manipulated to

$$\begin{aligned} 1 - e^{-j\alpha} &= e^{j\alpha}e^{-j\alpha} - e^{-j\alpha}, \\ &= e^{-j\alpha/2}\left(e^{j\alpha/2} - e^{-j\alpha/2}\right), \\ &= e^{-j\alpha/2}\left(2j\sin\frac{\alpha}{2}\right). \end{aligned} \tag{3.35}$$

Applying this to (3.34) gives the DTFT of a CIC filter as

$$\begin{aligned} H(\omega) &= \left(\frac{e^{-jRM\omega/2}\sin RM\omega/2}{e^{-j\omega/2}\sin\omega/2}\right)^N, \\ &= e^{jN(1-RM)\omega/2}\left(\frac{\sin RM\omega/2}{\sin\omega/2}\right)^N. \end{aligned} \tag{3.36}$$

By inspection, this has a magnitude response

$$|H(\omega)| = \left|\frac{\sin RM\omega/2}{\sin\omega/2}\right|^N, \tag{3.37}$$

and a linear phase response

$$\phi(\omega) = \frac{\omega N(1 - RM)}{2}. \tag{3.38}$$

The DTFT is a periodic function. When expressed in terms of ω, it has a period of 2π, while when expressed in terms of the normalised frequency f_n (that is, the frequency as a fraction of the sampling frequency f_s), it has a period of 1 [Ambardar 1999]. Hence, (3.37) and (3.38) can be expressed in terms of normalised frequency by substituting $\omega = 2\pi f_n$[4]:

$$|H(f_n)| = \left| \frac{\sin \pi RM f_n}{\sin \pi f_n} \right|^N, \tag{3.39}$$

$$\phi(f_n) = \pi f_n N(1 - RM). \tag{3.40}$$

It will be noted that the magnitude response is undefined whenever $f_n = k$ where k is any integer, i.e., at DC and multiples of the sampling frequency. To evaluate the magnitude response at these points, the limit of the response as f_n tends to k is taken and l'Hôpital's rule [l'Hôpital 1696] applied to give

$$\lim_{f \to k} |H(f)| = (RM)^N. \tag{3.41}$$

Comparing this to (3.28) shows that this increase in magnitude is equal to the maximum register growth of the filter. Hence this gain can be viewed as the signal being expanded to fit the range of the output number system. For this reason, the magnitude response of a CIC filter is often normalised to give

$$|H(f_n)| = \left| \frac{\sin \pi RM f_n}{RM \sin \pi f_n} \right|^N. \tag{3.42}$$

An example of this normalised magnitude response is shown in Figure 3.7.

[4]Note that the frequency is being normalised to the high sampling rate as is the convention throughout this thesis. Hogenauer [1981] normalises the frequency relative to the low sampling rate by substituting $\omega = 2\pi f_n / R$.

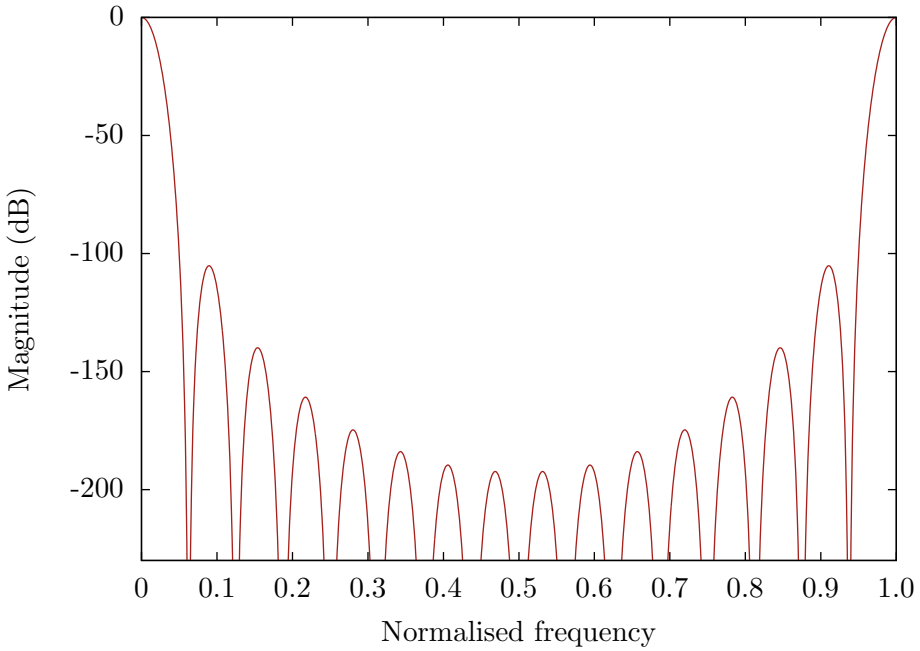


Figure 3.7: The normalised magnitude response of a CIC filter decimating a signal with $R = 16$, $M = 1$, and $N = 8$.

3.2.10.5 Compensation

As can be seen from Figure 3.7, the frequency response of a CIC filter has a significant droop in the passband. The typical solution is to use a compensation filter — placed after the CIC filter so it can operate at the slow sampling rate — to flatten the passband response. The ideal magnitude response of this compensation filter would be the inverse of the CIC magnitude response within the passband and zero elsewhere. Mathematically, this means

$$|H_{\text{comp}}(f_n)| = \begin{cases} \left| \frac{RM \sin \pi f_n}{\sin \pi RM f_n} \right|^N & f_n \text{ in passband,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.43}$$

Implementing this filter requires extra computational effort. However, the data output from this system will be post-processed later. This allows the compensation to be deferred to the post-processing stage rather than be performed in real time, and hence it can be ignored in the design of the decimation filter.

3.2.10.6 Selecting M and N

Assuming the decimation factor R is fixed, there are only two integer parameters that can be used to alter the response of a CIC filter: the number of stages N and the comb filter delay M. As can be seen from the magnitude response given in (3.42), increasing M results in more zeroes in the transfer function. Although having more zeroes increases the steepness of the transition band, it also increases the amount of droop in the passband as the zeroes will be located closer to the passband edge. M also features as a scaling factor in the denominator of the magnitude response, meaning increasing M decreases the gain of the filter. For these reasons, M is typically chosen to be either one or two.

N is therefore the main parameter chosen to set the attenuation of the filter. Rearranging the magnitude response from (3.42) gives the required number of stages to achieve the desired attenuation A (in decibels) at the normalised frequency f_n as

$$N = \left\lceil \frac{A}{20 \log_{10} \left| \frac{\sin \pi RM f_n}{RM \sin \pi f_n} \right|} \right\rceil \tag{3.44}$$

The compensation filter will be used to remove noise from the stopband, and hence the CIC filter only needs to prevent noise aliasing into the passband. When the downsampling is performed, the normalised frequencies $1/R, 2/R, \ldots, R$ alias to DC. From this, it follows that $1/R - f_{\text{pass}}$ will be the lowest frequency to alias back into the passband, and hence this is the frequency at which the attenuation needs to meet the design requirements. For the requirements given in Table 3.1, this normalised frequency is 0.0515. Applying (3.44) for the desired attenuation of 100 dB gives $N = 8$ if $M = 1$, or $N = 7$ if $M = 2$. Although the latter option requires one less stage, the amount of droop at the passband edge (12.9 dB) is significantly greater than the $M = 1$ option (3.6 dB), and hence the $M = 1$ option would be used.

A CIC filter requires N additions at the high sampling rate for the integrators, and N additions at the low sampling rate to implement the comb filters. When used to implement all 72 filters, this requires 6.12×10^9 additions per second, around 3.8 % of the additions needed for a standard FIR filter.

Filter	Computations per second	
	Additions	Multiplications
§3.2.5 Standard FIR	1.61×10^{11}	1.62×10^{11}
§3.2.6 Symmetric FIR	1.61×10^{11} (100 %)	8.14×10^{10} (50.2 %)
§3.2.7 Multistage FIR (two stages each with R=4)	3.82×10^{10} (23.7 %)	1.96×10^{10} (12.1 %)
§3.2.8 Half-band filter (four stages)	1.30×10^{10} (8.1 %)	7.20×10^{9} (4.4 %)
§3.2.8 16^{th}-band filter	1.52×10^{11} (94.4 %)	7.63×10^{10} (47.1 %)
§3.2.9 Polyphase filter (16 sub-filters)	1.14×10^{10} (7.1 %)	1.08×10^{10} (6.7 %)
§3.2.10 CIC filter (M=1, N=8)	6.12×10^{9} (3.8 %)	0

Table 3.2: The computation required to implement 72 decimation filters (i.e., one for each input channel of the KiwiSAS system) with the specifications from Table 3.1 using the architectures outlined in the preceding discussion. The numbers in brackets are the percentages of the computational effort required for a standard FIR filter.

3.2.11 Filter comparison

A comparison of the computational effort required for the various filters presented in this discussion is given in Table 3.2. As can be seen, the CIC filter requires just over half the number of additions per second of the polyphase filter which is the next most efficient. Even if a compensation filter needed to be implemented within this system, it is likely — due to the fact that it would operate at the slow sampling rate — that the CIC plus compensation combination would still require less additions per second. Even if this were not the case, the number of multiplications needed per second would certainly be much lower, and hence the CIC filter would still be preferable.

3.3 Transmitter

Due to losses in the transducers and the attenuation and spreading losses of acoustic energy in water, the KiwiSAS-IV system applies a 200 V_{p-p} signal to the transducers to obtain sufficient transmission range. The firmware outputs a sampled version of the desired waveform through a digital to analogue converter (DAC) to a class AB amplifier. A 12:120 turn step-up transformer is then used to generate the transmitted signal [Hayes 2003]. The amplifier is less than 50 % efficient, with the wasted energy

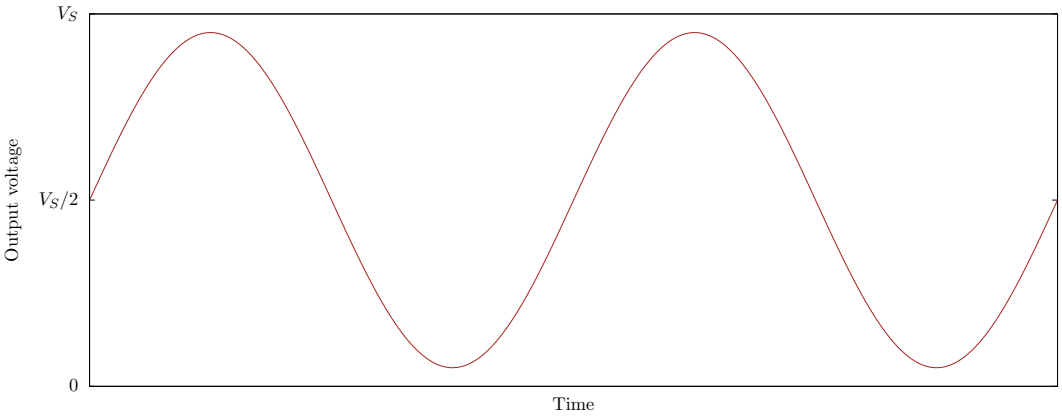
being dissipated as heat. This requires heatsinks to be used in order to prevent the system overheating. Along with the transformers, this takes up a significant amount of space. Replacing this arrangement with a class D amplifier will result in an increase in efficiency and a reduction in the heat produced. Additionally, the transformer is not required as a class D amplifier will be capable of directly generating the $200\,V_{p-p}$ signal, thus reducing the physical space occupied by the transmitter.

3.3.1 Class A, B and C amplifiers

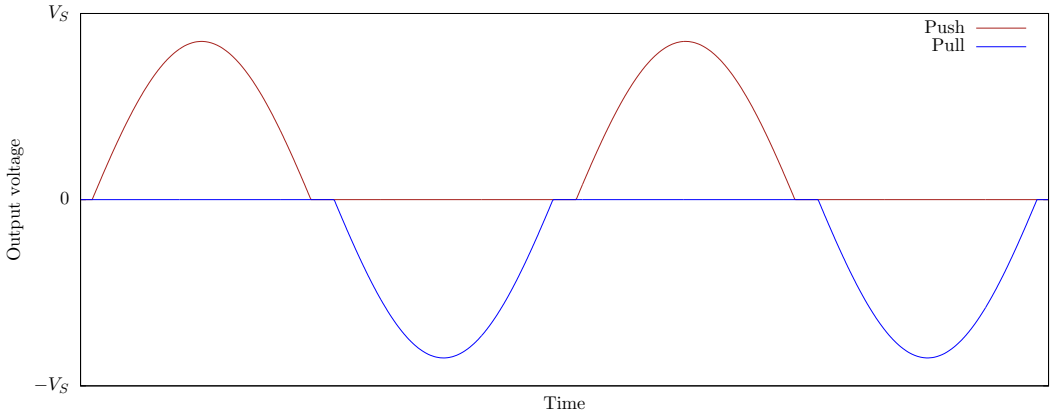
Class A, B, and C amplifiers can be broadly classified as 'linear' amplifiers in that they try to directly replicate the input signal by using transistors in their linear operating region. A class A amplifier uses a single transistor, resulting in a single supply circuit. As illustrated in Figure 3.8a, the transistor must be biased to mid-supply in order to allow for an equal signal swing in each direction. This biasing limits the efficiency of a class A amplifier; the theoretical maximum is 25 % if the load is series-fed or 50 % if transformer coupled [Boylestad and Nashelsky 2002], with actual efficiencies being lower than this.

Class B amplifiers operate without any biasing, meaning only half of the input signal — either the positive or negative half depending on the configuration — will be amplified with the transistor turned off for the other half. A resonant circuit can be placed at the output to recover the missing half of the circuit, albeit with some distortion of the signal [Hagen 1996]. A more common arrangement is to use two complementary class B amplifiers joined in a push-pull configuration, wherein one amplifier 'pushes' the output high during the positive half of the input and the other 'pulls' it low when the input is negative [Bowick 1997]. As shown in Figure 3.8b, when the input crosses zero both transistors are off leading to what is known as crossover distortion in the output. The maximum theoretical efficiency of a class B amplifier is 78.5 % [Boylestad and Nashelsky 2002], although as with the class A amplifier it is usually lower in practice due to losses in the transistors.

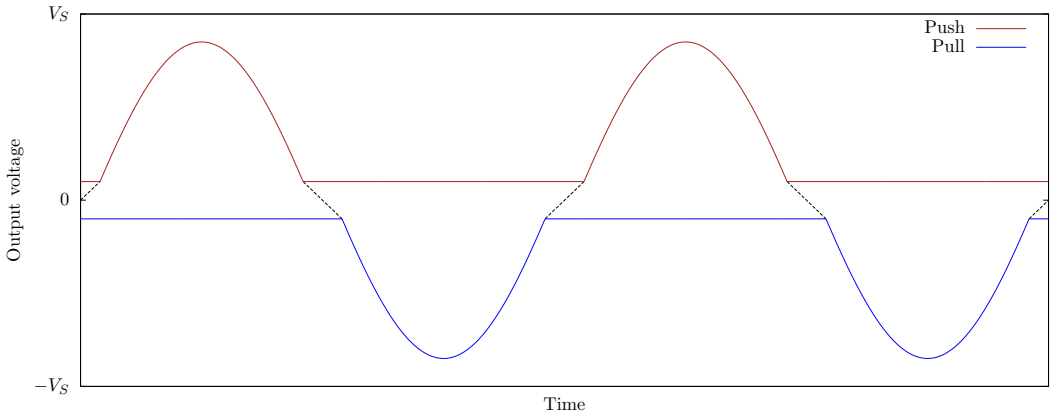
A commonly used trade-off is the class AB configuration. This uses a push-pull arrangement but applies a small bias as illustrated in Figure 3.8c. This has the affect



(a) Class A: Linear output but the necessary biasing reduces efficiency.



(b) Class B: No biasing needed but some non-linearity occurs when both transistors are off.



(c) Class AB: A small bias is applied to reduce non-linearity at the expense of efficiency.

Figure 3.8: A comparison of the operation of class A, class B and class AB amplifiers when a sinusoidal input is applied. The non-linearity is exaggerated for clarity.

of reducing the crossover distortion at the cost of decreased efficiency due to the power required to bias the transistors. Depending on the level of biasing, the efficiency can be anywhere between 25% and 78.5%.

A class C amplifier negatively biases the transistor so that it amplifies less than half of the input waveform; typically between 90° and 150° of a full cycle is reproduced [Frenzel 1997]. A tuned circuit is used in the output stage to recover the signal. Due to the distortion this creates, class C amplifiers are not suitable for audio circuits, but are commonly used in RF transmissions, including in transmission of FM signals which only vary in frequency [Rutledge 1999]. Providing it is known and repeatable, this distortion does not present a problem for the sonar system since a matched filter is used for echo detection. However, although the efficiency of a class C amplifier can approach 85 % in practice [Bowick 1997], it is still not as efficient as a class D amplifier.

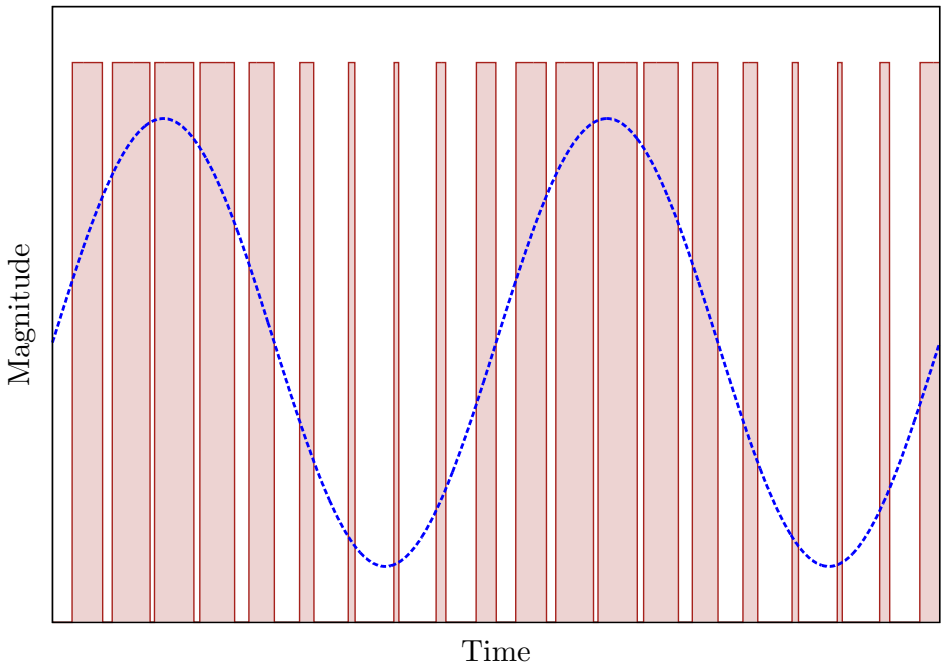
3.3.2 Class D amplifiers

Class D amplifiers are a type of switching amplifier, using the transistors (typically MOSFETs) as switches [Cox 2002]. They first appeared in the 1950s, but the limitations of both valves and early bipolar transistors meant they were not particularly practical until the appearance of power field effect transistors (FETs) [Self 2006]. The output of a class D amplifier is a square wave of a constant frequency, known as the switching frequency. The input signal is used to modulate the duty cycle of this square wave, making it a pulse-width modulation (PWM) signal. For a duty cycle $0 \leq D \leq 1$ the mean value of the output over one period T is

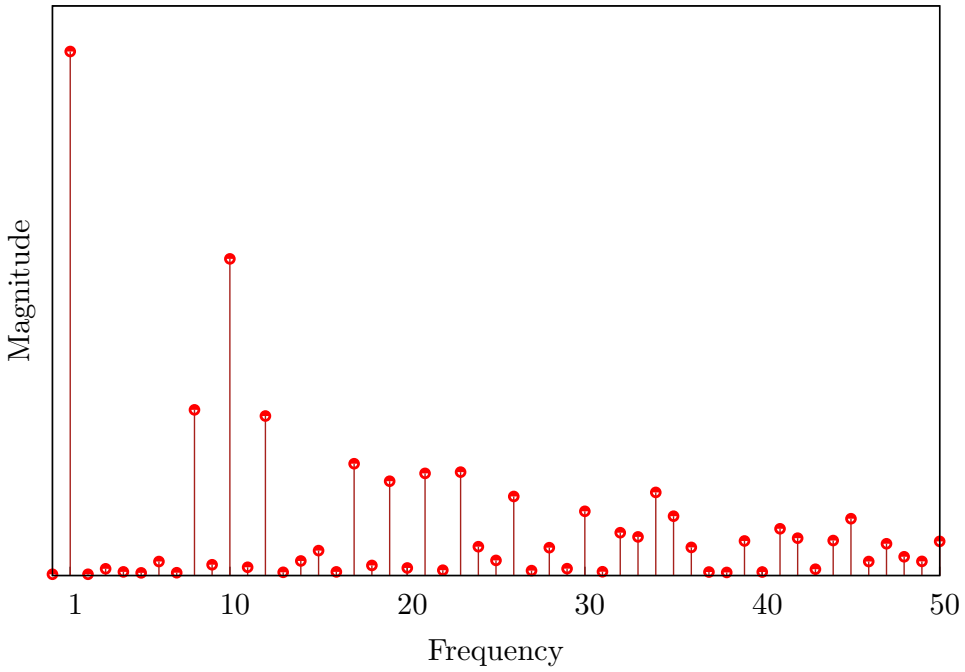
$$\begin{aligned} V_{\text{mean}} &= \frac{1}{T}\int_0^T V_{\text{out}}, \\ &= DV_H - (1-D)V_L, \end{aligned} \tag{3.45}$$

where V_H and V_L are the upper and lower voltages of the square wave respectively. In other words, the average value of the output is directly proportional to the duty cycle and hence the input signal. This means an amplified version of the input can be recovered by low-pass filtering.

An example of a class D output for a sinusoidal input is shown in Figure 3.9a.



(a) Time domain.



(b) Frequency domain harmonics.

Figure 3.9: The waveforms of a class D amplifier for a sinusoidal input. The switching frequency is ten times the frequency of the sinusoid.

As the amplitude of the input grows the pulses get wider and vice-versa. Taking the Fourier transform of this output, as illustrated in Figure 3.9b, clearly shows the desired signal and the switching along with the various harmonics.

In theory, the class D amplifier is 100% efficient as the transistors are either off or on, and, in an ideal MOSFET, neither of these states dissipate power. However, the on resistance in a real-world MOSFET causes some power loss, as does the leakage current present when it is off. Additionally, the requirement to charge or discharge the gate capacitance whenever the output switches further reduces the efficiency. Despite this, class D amplifiers can achieve over 90 % efficiency in practice [Boylestad and Nashelsky 2002], a large improvement over the class AB amplifier used in the existing sonar system.

3.4 T/R switch

With the same transducers being used for both transmitting and receiving, a T/R switch is required to protect the receiver circuits from the high voltages used in transmission. This T/R switch must limit the voltage presented to the AD9271 to within ±2 V, the maximum voltage the LNA inputs of the AD9271 are rated to, and be able to withstand the ±100 V voltages used in transmission. Shown in Figure 3.10, the T/R switch consists of four diodes in a bridge arrangement used to limit incoming signals to within one diode drop of ±5 V supplies, followed by a pair of Schottky diodes in a back-to-back arrangement to further clip the signals to ±300 mV. This design is taken from the AD9271 datasheet[5].

3.4.1 Operation

The operation of the T/R switch can be broken into three distinct parts. With no input voltage applied, the bridge diodes are turned on by the ±5 V supplies. Assuming the bridge is balanced (i.e., the diodes have the same forward voltage drops and the resistors have equal resistance), the input to the bridge will have a DC bias of 0 V. If

[5] Figure 46, 'AD9271 Octal LNA/VGA/AAF/ADC and Crosspoint Switch Data Sheet (Rev. B)', Analog Devices, May 2009. `http://www.analog.com/static/imported-files/data_sheets/AD9271.pdf`

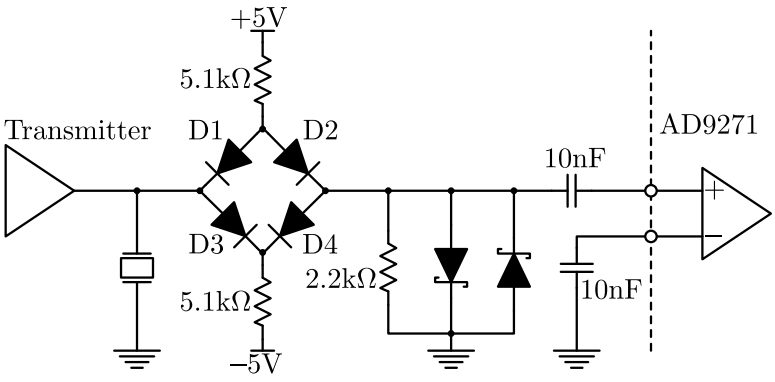
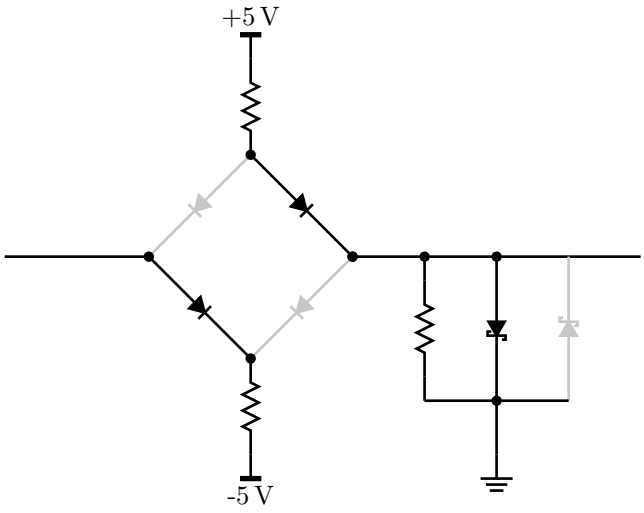


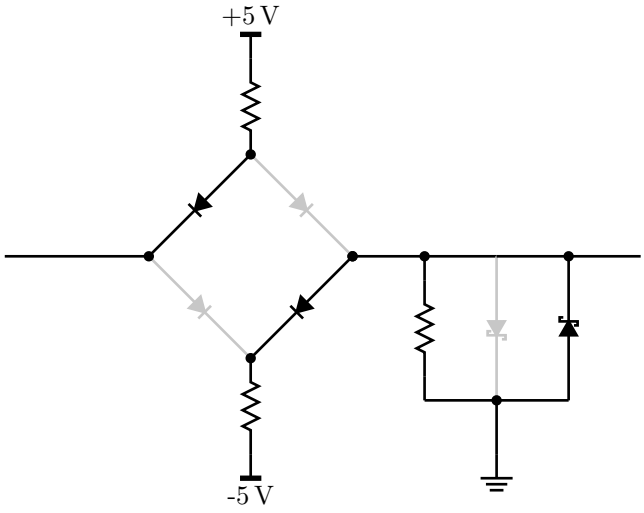
Figure 3.10: The T/R switch used to protect the receiver circuitry from the high voltage signals used for transmitting.

a small signal is then applied to the input, the diodes will 'float' with this input and allow it to pass through the bridge. If the magnitude of the signal is less than the voltage drop of the Schottky diodes (typically around 300 mV) then it will also pass this part of the switch and be AC-coupled into the LNA of the AD9271. However, if the magnitude of the signal is too large then one of the Schottky diodes will turn on, clipping the signal to ±300 mV.

The third mode of operation is when the input magnitude becomes larger than 5 V$-V_D$ (where V_D is the forward voltage drop of the bridge diodes). In this case, some of the bridge diodes are turned off. If the input is positive and large, then D1 will be reverse biased and hence turn off, while D3 will remain forward biased. The 2.2 kΩ resistor acts as a pull-down resistor to hold the output of the bridge near ground; this reverse biases D4 and forward biases D2. The output voltage of the bridge is then set by the voltage divider created by the 2.2 kΩ and 5.1 kΩ resistors; this also depends on the voltage drop caused by D2, but is typically around 1.3 V. This is large enough for the Schottky diodes to clip to 300 mV. This situation is illustrated in Figure 3.11a; the diode bridge isolates the receiver from the input and connects the input to the -5 V supply through the 5.1 kΩ resistor. With a 200 V signal, this corresponds to a current of approximately 40 mA flowing through this resistor. When the input is negative and large, the switch behaves similarly, connecting the input to the +5 V supply as shown in Figure 3.11b.



(a) Positive large voltages.



(b) Negative large voltages.

Figure 3.11: The state of the T/R switch diodes when blocking large voltages. The black diodes are conducting while the grey diodes are not.

CHAPTER 4

HARDWARE DESIGN

The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong, it usually turns out to be impossible to get at and repair.

Douglas Adams
Mostly Harmless

CHAPTER 4

HARDWARE DESIGN

This chapter describes the design and testing of a prototype circuit to test the system designed in Chapter 3. It contains eight identical channels, i.e., all the inputs of a single AD9271 were used, and was manufactured as a four layer printed circuit board (PCB). The selection of the components used to implement the transmitter is detailed, as is the decision to use an FPGA development board to control the circuitry. The layout of the PCB is described, and the results of testing the transmitter and T/R switch are presented.

4.1 Transmitter

The transmitter uses a class D amplifier switching between 0 V and 200 V to generate the signal fed to the transducer. Discrete MOSFETs are used for the switches with the PWM input being translated into the appropriate drive signals by a MOSFET driver IC. The 200 V supply is provided via an external connector for this prototype. In the final system a boost converter will be required to generate this voltage from the batteries of the sonar.

4.1.1 Class D amplifier

The on resistance of a P-channel MOSFET is larger than that of a similarly-sized N-channel MOSFET, typically by a factor of around three for silicon devices [Baker 2008]. This is due to the mobility of the electrons in the N-type material being greater than that of the holes in the P-type material. This disparity can be reduced by increasing the width of the channel in the P-channel MOSFET. However, this also results in

an increased gate capacitance, meaning the rise and fall times will be greater. For this reason, International Rectifier IRF7453 N-channel MOSFETs were used for both the low- and high-side switches in the class D amplifiers. Designed for high-frequency DC-DC converters, they are rated to handle a drain-to-source voltage of 250 V and a continuous drain current of 2.2 A. If driven with a gate voltage of 12 V — as is the intention in this design — they have a maximum on resistance of 200 mΩ.

An N-channel MOSFET requires a positive gate-to-source voltage to turn on, meaning the gate voltage of the high-side transistor must 'float' above the output of the amplifier to turn it on. Rather than manually implement this floating supply, the International Rectifier IRS20124S IC was selected to provide the drive signal for both MOSFETs. Although primarily designed for audio applications — where the maximum frequency is 20 kHz compared to the 110 kHz used by the sonar — the IRS20124S is capable of working with a switching frequency of up to 1 MHz. It contains an inbuilt boost converter, requiring only an external capacitor and diode, to generate the floating supply for the high-side MOSFET.

When the input switches from high to low or vice-versa, one of the MOSFETs must turn off and the other turn on. To remove any chance of both transistors being on simultaneously — which would short the 200 V supply to ground — a small delay is placed after the first transistor is turned off and before the second turns on; this delay is known as deadtime. The IRS20124 automatically enforces some deadtime when driving the MOSFETs.[1] The length of this deadtime can be varied in four 10 ns steps between 15 ns and 45 ns.

4.1.2 DC blocking

As the transducer is capacitive it has no DC value, meaning that any echo signals received will be superimposed on the DC bias at its terminals. When receiving signals, the T/R switch provides a 0 V DC bias at its input. However, the MOSFETs in the transmitter will both be off when the system is receiving, and hence their drain-to-source resistances will form a voltage divider between the 200 V supply and ground.

[1] 'IRS20124S Digital Audio Driver with Discrete Deadtime and Protection', International Rectifier, December 2006. `http://www.irf.com/product-info/datasheets/data/irs20124s.pdf`

Although the impedance of the T/R switch is much lower than the MOSFET resistances, and so the bias at the transducer should be 0 V, there may be an offset caused by the MOSFETs. A blocking capacitor is placed between the output of the class D amplifier and the transducer to prevent the DC bias from the MOSFETs reaching the transducer. A jumper is also provided to allow the blocking capacitor to be removed from the circuit for testing purposes.

Since the transducer is mainly capacitive, this arrangement forms a capacitive divider. The percentage of the amplifier output voltage which reaches the transducer is given by

$$\eta = \frac{C_B}{C_B + C_T} \times 100\%, \tag{4.1}$$

where C_B is the value of the blocking capacitor and C_T the capacitance of the transducer. To ensure the output voltage is not significantly reduced by this divider, the blocking capacitor should be much larger than the capacitance of the transducer. The bulk capacitance C_0 of the transducer BVD model developed in Section 2.4.2 is 630 pF, while the sum of all the capacitances in the model is 4.34 nF. Using these two values in (4.1) with the chosen blocking capacitor of 100 nF predicts that 99.4 % and 95.8 % of the amplifier output voltage will pass through to the transducer respectively.

4.1.3 Output filter

The velocity of the radiating face of the transducer is given by

$$U(f) = I(f)H(f), \tag{4.2}$$

where $I(f)$ the current the transducer draws and $H(f)$ its electroacoustic transfer function. Due to the resonant nature of the transducer, this means that the switching noise and harmonics of the amplifier output will be filtered before being transmitted into the water. However, as the transducer is connected between the amplifier and ground, the signal presented to the T/R switch is not filtered in this manner. Although the T/R switch should prevent the signal from reaching the AD9271, if the edges of the amplifier output are fast enough, they may exceed the maximum input rating of the

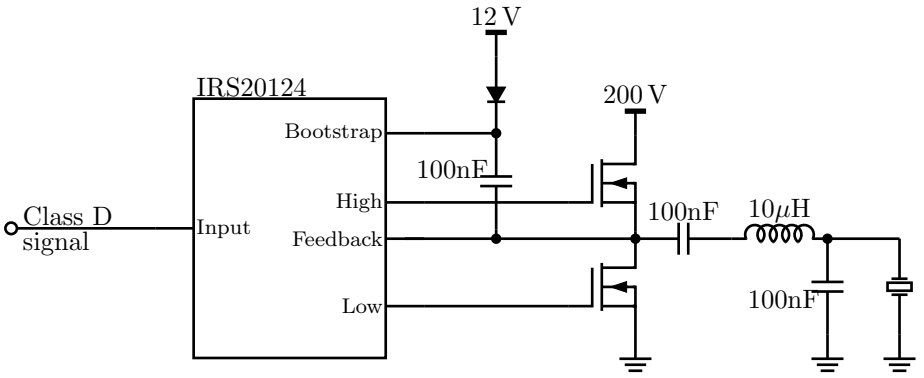


Figure 4.1: The transmitter circuit used in the prototype, comprising a MOSFET driver, two N-channel MOSFETs, a DC blocking capacitor, and a second order LC low-pass filter. Both the blocking capacitor and filter can be removed from the circuit by jumpers.

AD9271 before the T/R switch can react. If this is the case, then a low-pass filter needs to be used to slow down or remove the edges prior to the T/R switch.

To this end, a second-order low-pass filter was placed between the DC blocking capacitor and the transducer. As with the blocking capacitor, jumpers are provided so that the filter can be removed from the circuit for testing. An LC filter was used so that the filtered energy is stored rather than lost as heat, thus minimising the loss of efficiency caused by the filtering. With a 10 μH inductor and a 100 nF capacitor, the filter has a -3 dB (or half-power) point at 159 kHz, sufficiently above the highest frequency (110 kHz) of the transmitted signal to prevent distorting it.

4.2 FPGA

Using a microprocessor-based system to receive the sampled data from the AD9271 is not feasible due to the data rate. Even at the slowest sampling rate of 10 MHz a single channel outputs data at 120 Mbit/s. Instead an FPGA is used, allowing the processing for each channel to be pipelined; additionally each channel can be handled in parallel. After decimation, any desired further processing can be performed and the data then passed to the back-end system for storage.

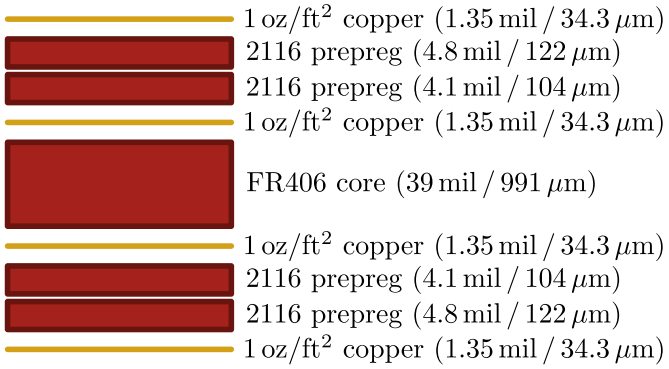


Figure 4.2: The stackup of the manufactured PCB with the thickness of each layer indicated. The thickness of the final board is 62.2 mil (1.58 mm) with a ±10 % manufacturing tolerance.

4.2.1 Spartan-3E starter kit

Rather than integrate an FPGA into the prototype circuit, an FPGA development board performs the processing. The chosen board is a Digilent Spartan-3E starter kit, based upon the Xilinx Spartan-3E XCS500E FPGA. It contains a USB programming interface along with various external connectors. The front-end prototype PCB is connected via a 100-pin expansion header which provides two power supplies as well as connections to 43 pins on the FPGA.

4.3 PCB design

Both the schematics and the PCB layout for the prototype were created in Altium Designer 6 and are shown in Appendix A and Appendix B respectively. Due to the repetitive nature of the transmitter and T/R switch circuits, one copy of each circuit was created and then reused to form all eight channels. The prototype PCB was manufactured by Advanced Circuits, a United States of America-based company. Their four layer prototype service was selected, with the resulting PCB having the stackup shown in Figure 4.2.

4.3.1 LVDS traces

LVDS is a current loop system. The driver outputs a small current — nominally 3.5 mA in the case of the AD9271 — which appears as a voltage swing across the termination resistor at the receiver. The common mode voltage of an LVDS link is 1.25 V and the differential voltage is 350 mV. To achieve this swing with the AD9271 output, a 100 Ω termination resistor is required. The Spartan-3E starter kit has pads available for termination resistors on some of the input pins, while the FPGA itself has an on-chip termination circuit which can be configured to behave like a 120 Ω termination resistor. Since not enough pads were available to handle all the LVDS inputs from the AD9271, it was decided to solely use the FPGA's internal termination.

4.3.1.1 Trace width

In order to minimise signal reflections between the LVDS traces and the termination circuit, their impedances should be matched. Due to the differential nature of the termination, the characteristic impedance of the traces should be half the value of the termination resistor [Ritchey 1999], i.e., 60 Ω. The characteristic impedance Z_0 of a microstrip trace is given by Dworsky [1979] as

$$Z_0 = \frac{87}{\sqrt{\epsilon_r + 1.41}} \ln\left(\frac{5.98H}{0.8W + T}\right), \tag{4.3}$$

where ϵ_r is the relative permittivity of the dielectric material, H the separation between the trace and the ground plane, W the width of the trace, and T the thickness of the trace. Since ϵ_r, H, and T are all determined by the manufacturing process, the impedance must be controlled by the width of the trace. Rearranging (4.3) gives the necessary width for a given impedance as

$$W = \frac{7.475H}{\exp\left(\frac{Z_0\sqrt{\epsilon_r+1.41}}{87}\right)} - 1.25T. \tag{4.4}$$

Since the LVDS traces will be on the top layer and the ground plane on the first internal layer, the distance between them is $H = 226\,\mu$m. Although Advanced Circuits

do not specify who manufactures the 2116 prepreg they use, a typical piece of 2116 has a relative permittivity $\epsilon_r \approx 4.5$. Finally, assuming the thickness of the copper is not altered during manufacturing, the thickness of the trace will be $T = 34.3\,\mu$m. Using (4.4) then gives the width of the trace as 273μm or 10.75 mil for a $60\,\Omega$ characteristic impedance.

4.3.1.2 Length mismatches

An LVDS receiver is essentially a crossing detector. Providing the slopes of the positive and complementary signals cross when the value changes, the transition will be correctly detected. This means the propagation time of the two traces in a differential pair must be within two rise or fall times of each other [Ritchey 1999]. The rise and fall times of the AD9271 LVDS outputs are both 300 ps, meaning the propagation of the signals cannot differ by more than 600 ps. The speed of propagation in a microstrip trace is given by

$$v = \frac{4.35 \times 10^8}{\sqrt{\epsilon_r + 1.41}}. \tag{4.5}$$

With the dielectric material having a relative permittivity of 4.5, the LVDS signals will travel along the trace at $1.79 \times 10^8\,\mathrm{ms}^{-1}$, or 107 mm every 600 ps. This means the two traces in an LVDS differential pair could differ by around 100 mm and the signal would still be correctly received. To allow for any mismatches in the starter kit and give a suitable margin for issues such as a different propagation speed and receiver error, the traces on the prototype board were matched to within 2.2 mm.

The data clock output by the AD9271 toggles halfway through each bit period with a margin of error of 300 ps. A length mismatch between the clock and data lines could cause the clock edge to be received during a different bit period, meaning that the data read by the system would be incorrect. At the maximum data rate of 600 Mbps, a single bit period is 1.67 ns, giving the maximum allowable difference in propagation times between the clock and data traces as 535 ps. At the propagation speed calculated from (4.5), this corresponds to a difference in length of 95.8 mm. As before, the clock-to-data mismatches on the prototype PCB were kept lower than this to give a margin for error. The largest difference in length between the clock line and a data line is

6.1 mm.

4.3.2 Layout

The prototype PCB uses four layers. The top layer contains the majority of the components and the receiver traces, the first internal layer is dedicated to ground planes, the second internal layer to power planes for the various voltages required, and the bottom layer contains some components and the transmitter traces. There are three ground planes, one each for the analogue, digital and high voltage areas of the design. They are physically separated and joined by 0 Ω resistors at star points to avoid noise from the high voltage and digital sections being coupled into the analogue components.

The connector for the FPGA development board is placed at the left-hand side of the PCB with the AD9271 next to it. The LVDS traces are laid out on the top along with the serial peripheral interface (SPI) lines used to control the AD9271. The transmitters and T/R switches for all eight channels are placed in a column at the right-hand edge of the PCB. All eight channels are laid out identically, with the transmitter circuitry at the right, the connector to the transducer in the middle and the T/R switch towards the middle of the board. Jumpers are provided to allow both the DC blocking capacitor and low-pass filter to be removed from the transmitter circuit for testing purposes. The outputs of the T/R switches are AC-coupled into the inputs of the AD9271.

The starter kit provides a +5 V supply over the expansion header, along with the BANK0 rail (the supply voltage used for the IO pins on the FPGA) which can be either 2.5 V or 3.3 V depending on the configuration of the starter kit. Two sets of voltage regulators are used on the prototype board, one for the digital portion of the AD9271 and the other the analogue portion, with each requiring a 1.8 V supply. Each set contains both a switching regulator and a linear regulator; which one is used to regulate the voltage is selected via a jumper. This allows the noise levels of both regulators to be tested.

The T/R switches also require a -5 V supply, and the transmitter needs both 12 V and 200 V for operation. For this prototype these voltages are supplied via connectors, all placed at the top of the PCB above the circuitry for each channel. Connectors are

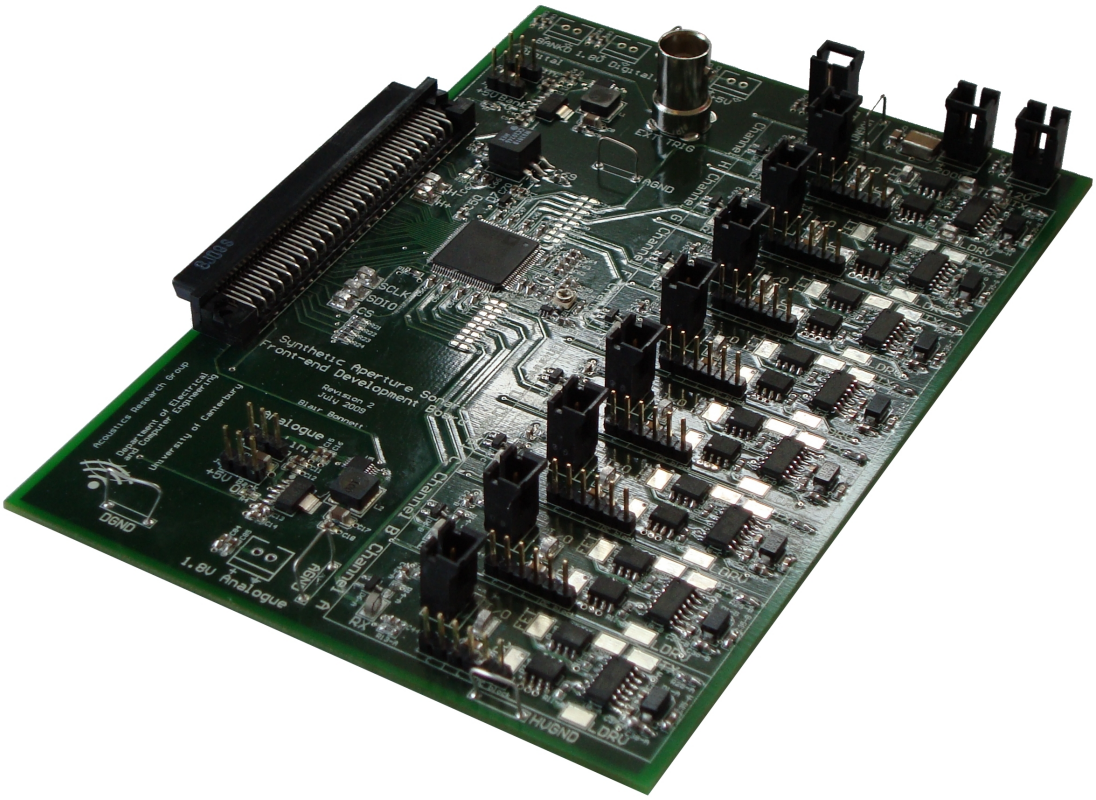


Figure 4.3: The populated prototype PCB.

also available for all the other supplies for testing or measurement purposes.

The full layout of the PCB is available in Appendix B, and a photograph of the populated PCB is shown in Figure 4.3.

4.4 Hardware testing

Once the PCB had been manufactured and populated, the transmitter circuits were tested. This was performed stage by stage, starting with the class D amplifier by itself. The effect of the blocking capacitor and the operation of the T/R switch were then investigated. Finally, the LC low-pass filter was tested.

4.4.1 Amplifier

The first tests of the class D amplifier were completed with no load. A 800 kHz square wave with a 50 % percent duty cycle was applied to the input. The output of the

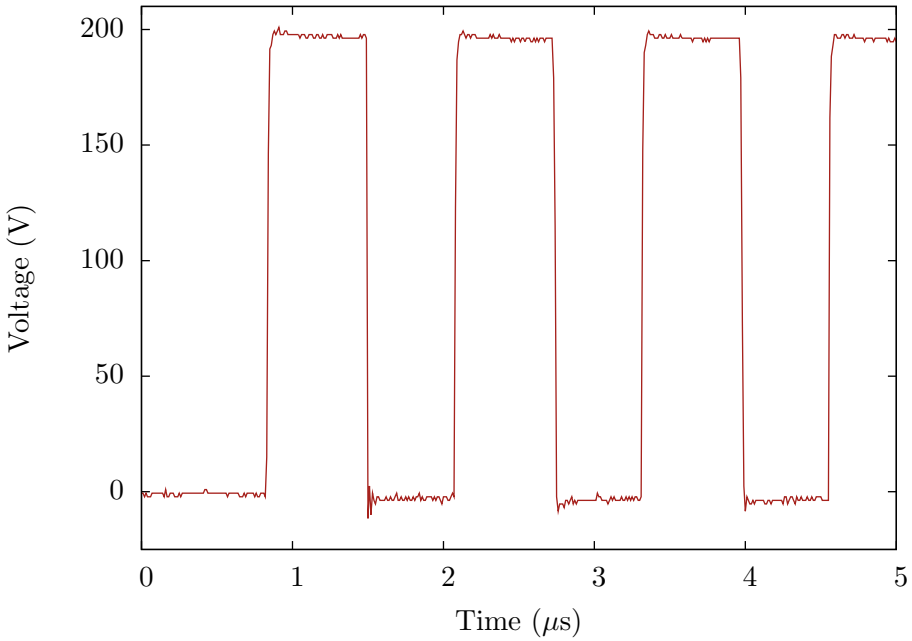


Figure 4.4: The output from the class D amplifier with no load and an 800 kHz, 50 % duty cycle input.

amplifier is shown in Figure 4.4 and reveals a fairly clean waveform with minimal overshoot and occasional short-term ringing. Figure 4.5 zooms in on the rising and falling edges of the waveform. From this, the 10 %-to-90 % rise and fall times can be measured as 10 ns and 5 ns respectively. These correspond to 0.8 % and 0.4 % of the output period (1.25 μs).

As shown in Figure 4.6, there is a delay of around 120 ns between the input switching and the output reacting, caused by the propagation delay of the MOSFET driver. The initial overshoot on the input is caused by the signal being reflected, indicating that termination is required on the amplifier input. The subsequent ringing on the input is caused by noise from the output switching.

As mentioned previously, the driver enforces deadtime between turning one transistor off and the other one on in order to prevent the 200 V supply shorting to ground. Figure 4.7 shows the signals driving the two MOSFETs and marks the periods when both transistors are fully off, indicating that the deadtime is being correctly inserted into the waveform.

A spare transducer was then connected to the class D amplifier to see what affect

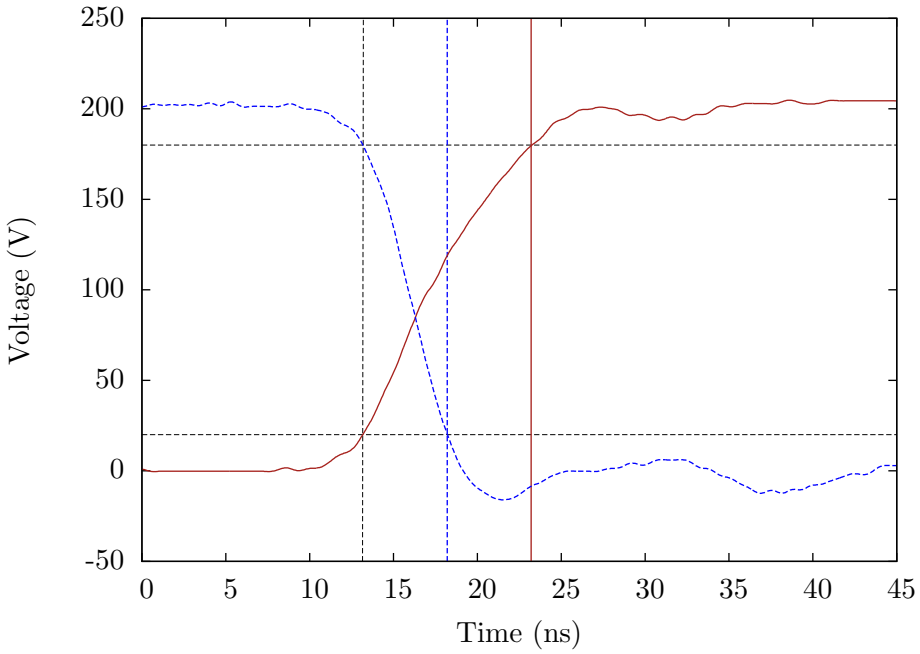


Figure 4.5: A closeup of the rising and falling edges of the class D amplifier with no load. The 10 %-to-90 % rise and fall times are marked.

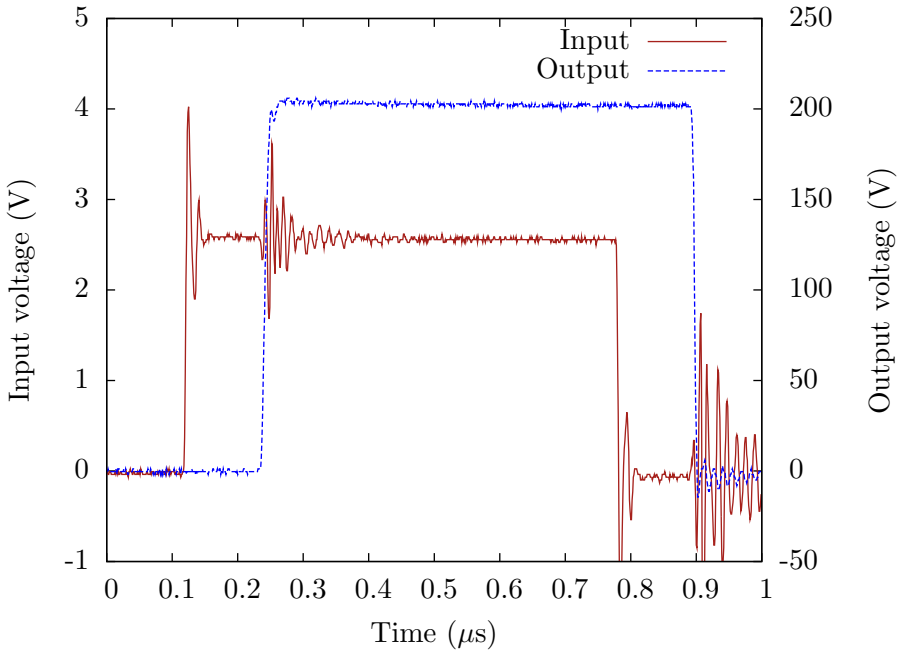


Figure 4.6: The delay between the input to the amplifier switching and the output reacting.

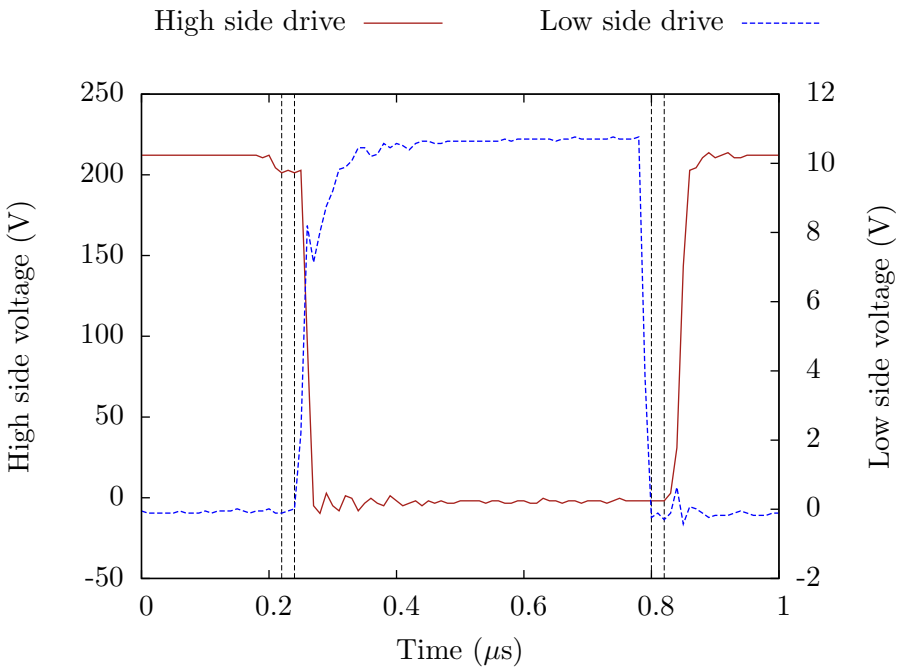


Figure 4.7: The signals driving the MOSFETs in the class D amplifier. The vertical lines mark the deadtime — the periods when both transistors are fully off.

this had on the output waveforms. As Figure 4.8 shows, this lead to more ringing in the waveform. The capacitance of the transducer also reduced the rise and fall times of the output. As illustrated in Figure 4.9, the 10 %-to-90 % rise time is now 12.5 ns, and the corresponding fall time is 6.5 ns. As the loading does not effect the driver, the propagation delay and deadtime remained unchanged with the addition of the transducer.

4.4.2 Blocking capacitor

The DC blocking capacitor was then inserted into the circuit. As shown in Figure 4.10, the output has shifted downwards by around -15 V and the peak-to-peak voltage is not noticeably affected by the presence of the capacitor. The rising and falling edges of the output are given in Figure 4.11. The fall time has worsened to 7.3 ns, but the capacitor has improved the rise time to 6.25 ns.

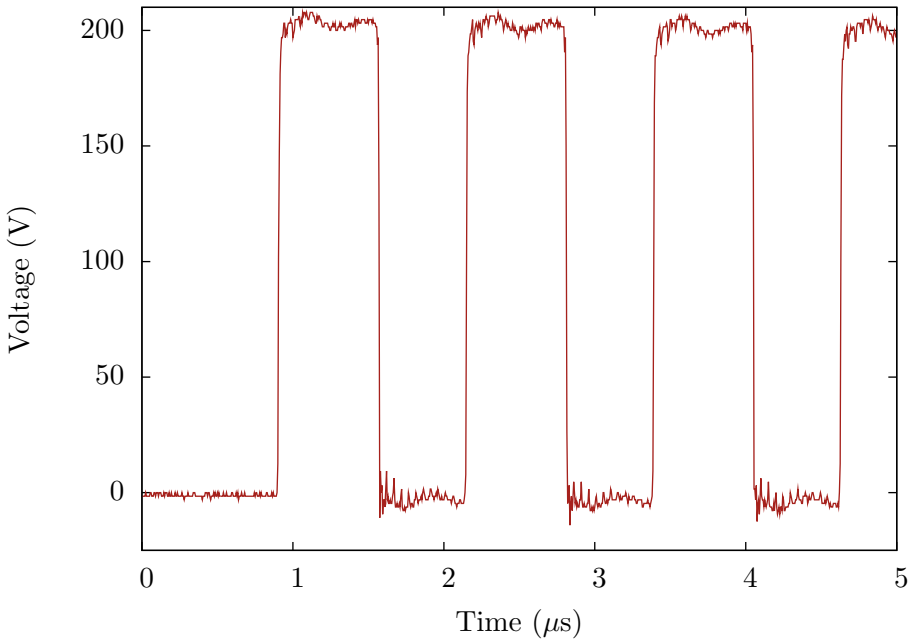


Figure 4.8: The output from the class D amplifier driving a transducer with a 800 kHz, 50 % duty cycle input.

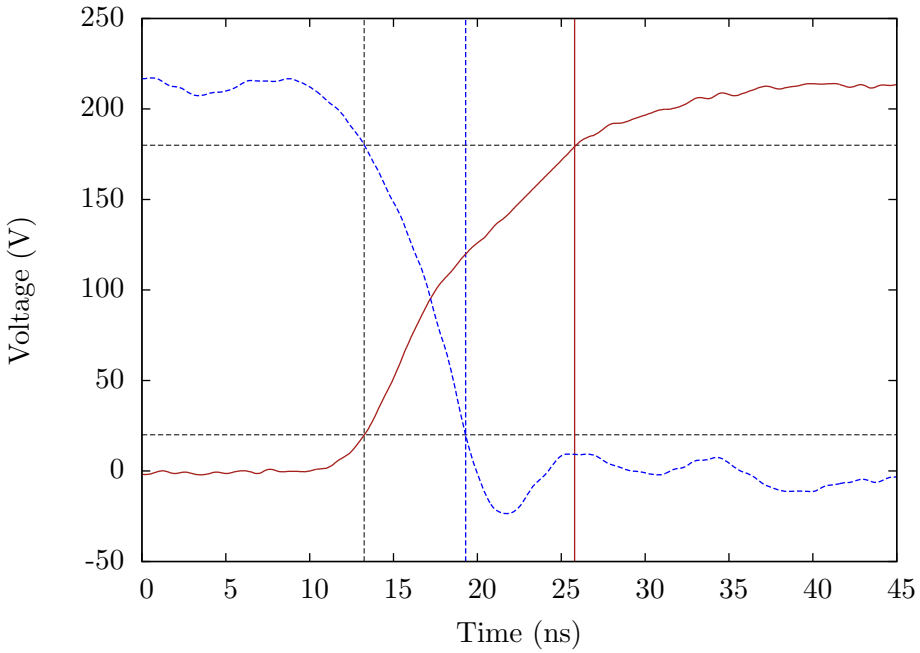


Figure 4.9: A closeup of the rising and falling edges of the class D amplifier when driving a transducer with the 10 %-to-90 % rise and fall times marked.

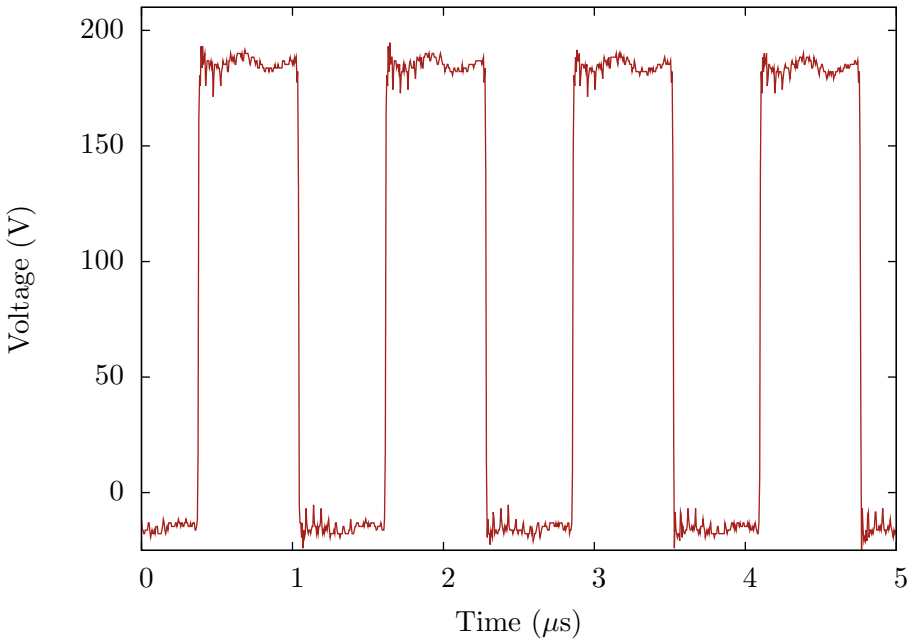


Figure 4.10: The output from the class D amplifier driving a transducer through the blocking capacitor with a 800 kHz, 50 % duty cycle input.

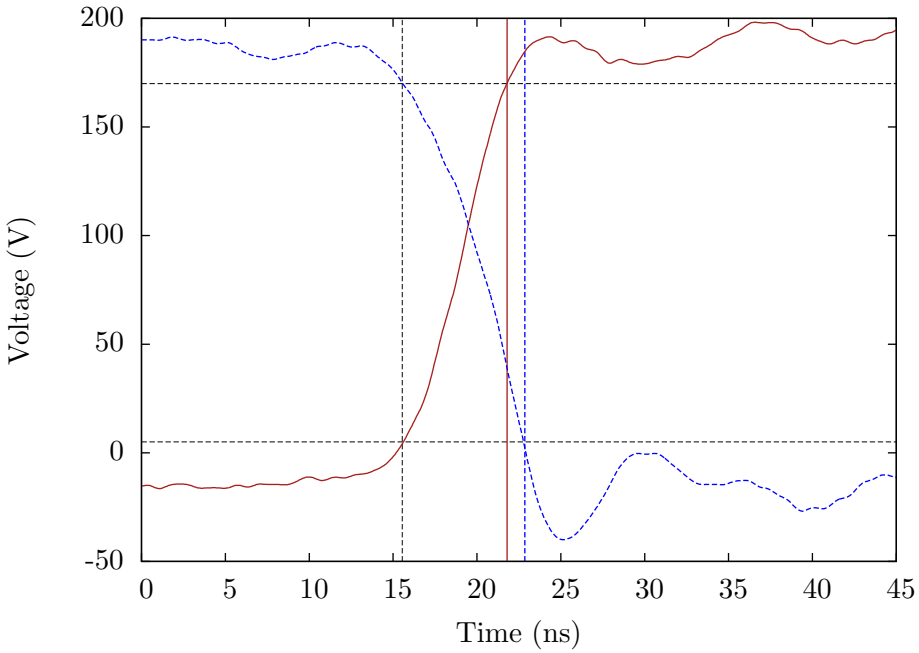


Figure 4.11: A closeup of the rising and falling edges of the waveform applied to a transducer by the class D amplifier and blocking capacitor. The 10 %-to-90 % rise and fall times are marked.

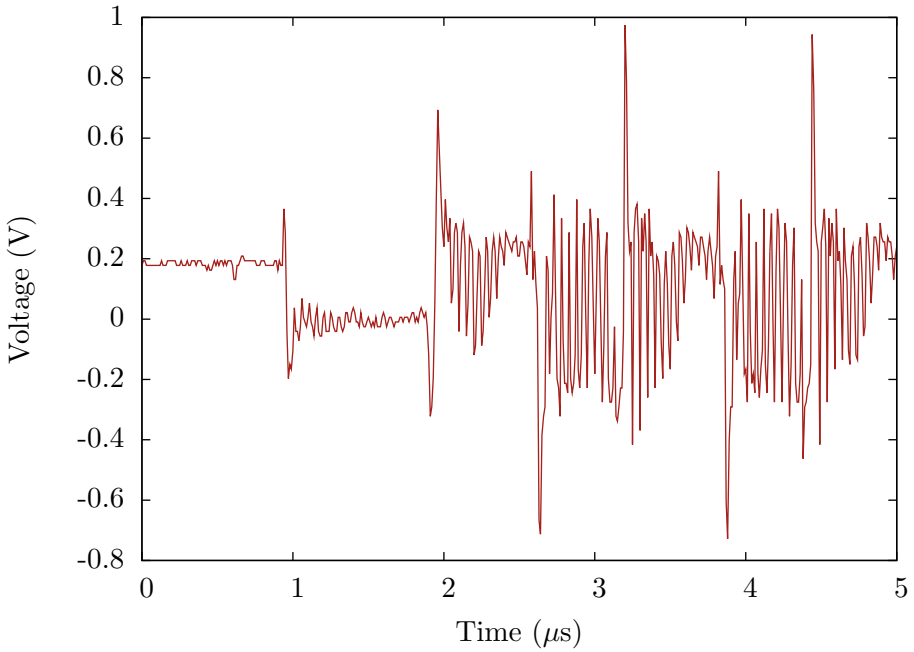


Figure 4.12: The output of the T/R switch when the class D amplifier drives a transducer from an 800 kHz, 50 % duty cycle input.

4.4.3 T/R switch

The output of the T/R switch was investigated with and without the blocking capacitor. Without the blocking capacitor (Figure 4.12), there is a DC bias of 200 mV at the output of the T/R switch. This shows that the voltage divider created by the drain-to-source resistances of the MOSFETs is not completely negated by the 0 V bias of the T/R switch input. As the Schottky diodes in the T/R switch start clipping incoming signals at ±300 mV, this bias would mean that any received signal above 100 mV would be clipped. The spikes caused by the leading edges of the transmitted waveform are limited to ±1 V, within the maximum rating of the AD9271 inputs.

Adding the blocking capacitor to the circuit removes the bias as shown in Figure 4.13. In this case, the magnitude of the spikes has increased slightly, but they are still within the maximum ratings.

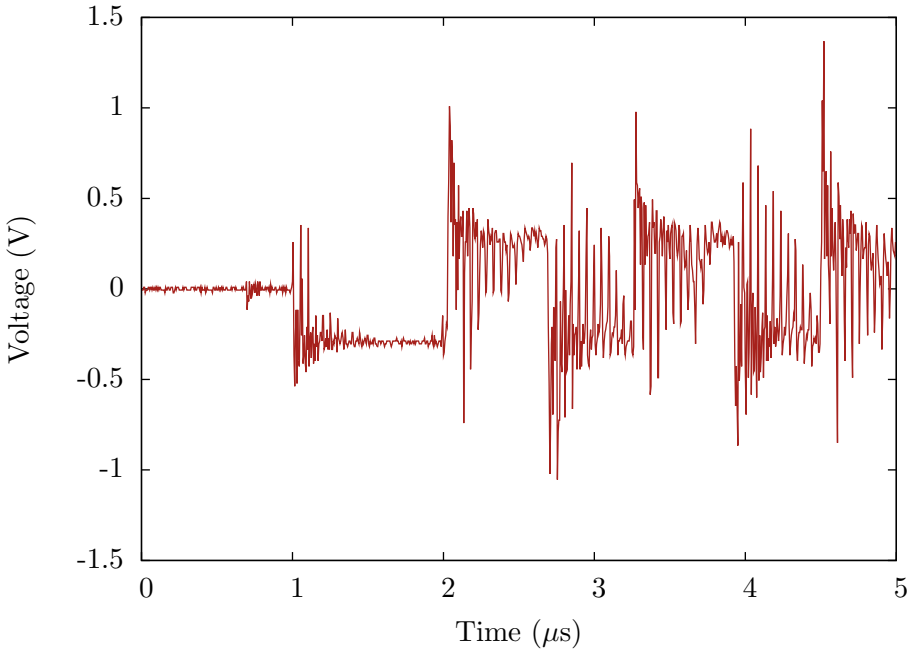


Figure 4.13: The output of the T/R switch when the class D amplifier drives a transducer with an 800 kHz signal through the blocking capacitor.

4.4.4 Low-pass filter

Although the T/R switch satisfactorily blocked the transmitted waveform, the low-pass LC filter was then inserted into the system. At low supply voltages (below around 30 V), this successfully filtered the waveform applied to the transducer and removed the spikes at the output of the T/R switch. However, when the supply voltage was increased to the full 200 V, the high-side transistors were destroyed. This suggests that the filter is providing a low impedance impedance path to ground, causing the 200 V supply to be shorted and a large current to flow through the transistor. This occurred both with and without the blocking capacitor present, eliminating the possibility that the filter and blocking capacitor were resonating. It is assumed that the filter is responding to the switching frequency or one of the harmonics in the output, despite the fact that its resonant frequency is well below these frequencies. However, as the filter is not required for the T/R switch to block the transmitted signal, this was not investigated further.

CHAPTER 5

FIRMWARE DEVELOPMENT

Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.

Rich Cook
The Wizardry Compiled

CHAPTER 5

FIRMWARE DEVELOPMENT

This chapter describes the development and testing of the firmware for the FPGA to control the hardware on the prototype front-end. This was written in VHDL and created in a modular fashion, with each section of the processing handled by separate modules. These are then connected together to implement the complete system. The open-source GHDL simulator was employed to verify the operation of each module, and version 9.2i of the Xilinx ISE toolchain was used to synthesize the VHDL into the bitstreams used to program the FPGA.

5.1 Module design

As the back-end system for the sonar had not yet been developed, the FPGA development board was used to provide control inputs to the firmware and collect the output data. This necessitated the development of two 'helper' modules — a button debouncer and an RS-232 data collection module. The transmitter firmware was developed as a single module, while the receiver was split into three modules: a SPI communication module, a deserialiser, and a CIC filter module. The full VHDL code for all of the modules described in this section is available in Appendix C.

5.1.1 Helper modules

The FPGA development board contains a number of slide switches and pushbuttons which were used to provide enable and trigger signals. However, there is no debouncing circuitry on the board, meaning that the signal provided to the FPGA from the switches is noisy. To counter this, a debouncer was implemented in firmware. The

development board also provides a DB-9 connector for RS-232 communications along with a level translator to convert between the output voltage of the FPGA and the necessary voltages for RS-232. A module was developed to capture a set of data and transmit it over this RS-232 link.

5.1.1.1 Debouncer

The input to the debouncing module is read on the rising edge of a sampling clock, and then passed through a single-tap low-pass IIR filter with the difference equation

$$y[n] = 0.75y[n-1] + x[n]. \tag{5.1}$$

The filter is implemented using unsigned eight bit numbers with a low input mapped to $x[n] = 0$ and a high input mapped to $x[n] = 63_{10}$. Simulation of the module confirmed that these values prevented the filter overflowing.

The output of the filter is then passed through a Schmitt trigger to remove any remaining noise. The filter must go above 240_{10} for the output of the debouncer to transition from low to high, and for a high-to-low transition the filter output must drop below 15_{10}. These thresholds require ten clock cycles for a completely clean signal to change the output of the module, with a noisy signal taking longer. As with the input mapping, they were chosen through simulation.

5.1.1.2 RS-232 data collection

A data collection module of some form was required in order to transmit the received data back to a computer for checking. Although the development board has an Ethernet connector, implementing the necessary firmware to support Ethernet communication solely for testing purposes is overly complex for the timeframe of this project. Instead, an RS-232 link was used to transmit the data. However, RS-232 communication is often limited to 115.2 kbps, the maximum speed supported by most receivers. Even with the minimum amount of overhead this equates to a data rate of 92.16 kbps.

To overcome the problems caused by this low data rate, the module incorporates a data buffer formed from the block random access memory (RAM) available in the

FPGA. This can be written to at speeds in excess of 200 MHz, sufficient to handle even the highest sampling rate of 50 MSPS. The data collection module uses this RAM as a circular buffer to store incoming data. When the buffer is full, any incoming data is blocked so that the data already in the buffer is not overwritten. The module automatically transmits data across the link whenever the buffer is not empty.

The module is designed to be highly flexible, with both the width of the input data and the depth of the buffer configurable when instantiating the module. Similarly, the speed of the RS-232 link can be configured, as can the format of the data — the number of data bits, the number of stop bits, and whether or not to use parity bits for error detection. If the width of the input data is greater than the number of data bits per transmission, the module splits data across multiple transmissions.

5.1.2 Transmitter

The switching frequency of the class D amplifier was selected as 800 kHz, over seven times the highest frequency in the transmitted signal. This allows a suitable frequency margin for the LC low-pass filter in the transmitter circuitry to remove many of the harmonics. Since the waveform is a square wave, it has only odd harmonics. The third harmonic will be at 2.4 MHz and the fifth at 4 MHz, meaning that any energy from the fundamental or the harmonics that gets through the LC filter will avoid resonances in the transducer response.

As the transmitted signal rarely changes, it is stored in memory as a series of duty cycles. The transmitter module then reads the values and converts them into the corresponding corresponding PWM waveform to be output to the MOSFET driver. With the clock frequency on the development board being 50 MHz, the output is updated 63 times per PWM period, meaning the resolution of the duty cycle is 1.59 %. With the duty cycles being stored using eight bits each, the 12.5 ms ping used in the current system would require 10 kB of memory.

The desired ping repetition rate is set when instantiating the transmitter module. At the start of each ping, the module starts reading the duty cycle values from memory. As the memory interface on the final system will probably differ from that available on

the development board, a generic interface is used. A clock signal is provided, with the memory required to put the next duty cycle value on the data bus on the rising edge of the clock. When the memory indicates that the last duty cycle has been given, the transmitter is shut down to allow the receiver to capture the returning signal. After the appropriate delay, the next ping is then begun.

The conversion of the duty cycle into a PWM waveform is straightforward. A counter is incremented at the main clock frequency and reset at the beginning of each PWM period. A comparator is then used to generate the output; if the counter is below the desired duty cycle value, the output is high and vice-versa.

5.1.3 Receiver

The receiver is broken into a number of modules. The AD9271 has an SPI interface which allows properties such as the LNA gain to be set; to utilise this, a SPI communications module was created. A deserialiser module was designed to read the DDR serial signals returned by the AD9271 and convert them to parallel values, and a CIC filter module was written to decimate the received data.

5.1.3.1 AD9271 SPI configuration

The AD9271 has a number of registers which configure its operation. The values of these registers are able to be set or read via an SPI interface so a VHDL module was created to simplify this configuration process. When writing to a register, the module takes the address of the register and the value to write to it. This is then formatted into the appropriate serial datastream and sent to the AD9271. Similarly, when reading the value of a register, it takes the address of the register and sends the necessary command to the AD9271. The returned value is then deserialised and presented at the output of the module.

5.1.3.2 Deserialiser

The first step in processing the data returned by the AD9271 is to deserialise it. The serial data is returned most significant bit first using DDR transfers, i.e., on both edges

of the data clock. The start of each piece of data is marked by the rising edge of the frame clock. Two six-bit shift registers are used to capture the incoming data, one driven by the rising edge of the data clock and the other by the falling edge. The outputs of these shift registers are interleaved and fed to the input of a twelve-bit register; this register is latched on the rising edge of the frame clock to store the parallel value for further use. The module contains an active-high reset input to reset both the shift registers and the output register when desired.

5.1.3.3 CIC filter

After the data has been deserialised, it will be decimated using a CIC filter. Rather than hard-coding the filter properties (the rate change factor R, number of delay stages in the comb section M, and number of integrator and comb sections N), they are specified when instantiating the module. The VHDL `generate` construct is then utilised to create the appropriate number of stages. Additionally, the desired input data width and output data width can be specified on instantiation. The module calculates the number of extra bits that will be generated by the filter, and the output is then either truncated or sign-extended to fit the requested width. An active-high reset line is available to clear the internal registers of the filter. The operation of this module was checked both by simulation and by testing on the FPGA development board. In both cases, a series of known values were fed to the input of the filter, and the outputs collected and verified to be the correct values. This checking was performed with a variety of different filter parameters.

5.2 Receiver verification

The AD9271 contains a number of test modes which are activated via the SPI interface and designed to check whether a receiver is operating correctly or not. One of these outputs a pseudorandom sequence rather than the sampled data. The sequence is generated using a 23-bit shift register with the feedback polynomial

$$x^{23} + x^{18} + 1, \tag{5.2}$$

i.e., the next input is formed from the modulo-2 addition of the current 18^{th} and 23^{rd} bits[1]. Twelve succesive outputs of the shift register are used to form each value returned by the AD9271. With the AD9271 placed in this test mode, the outputs of the receiver firmware were captured. By comparing the captured values to the expected values, it was confirmed that the receiver firmware was operating correctly.

5.3 Noise characterisation

Any noise introduced by the receiver will make the echoes harder to detect and degrade the performance of the sonar. Once the receiver firmware was written and tested, the noise levels of the AD9271 and the T/R switch were investigated. The CIC filter was also tested to check what effect the extra resolution gained by decimation had on the noise performance of the receiver.

5.3.1 AD9271

The datasheet of the AD9271 presents a histogram of the sample codes output by the chip when its inputs are short-circuited[2]. This gives an indication of the noise introduced by the conversion process. To compare the performance of the prototype front-end to the manufacturer's specifications, the inputs to one of the channels of the AD9271 was short-circuited and just over one million samples of the output were collected. The corresponding histogram in shown in Figure 5.1a and has a mean of -13.3 and a variance of 2.2. Compared to the histogram in the datasheet the outer bins are slightly larger and there is an offset of -13. This offset was present on all the channels, suggesting it was caused by a process such as thermal drift or a reference error. The power spectral density (PSD) of the samples was calculated by dividing them into 32 groups, taking the discrete Fourier transform (DFT) of each group and then averaging the squared magnitudes of the DFTs. As shown in Figure 5.1b, the PSD indicates the noise floor of the samples is around -70 dB, slightly above the 74 dB that (3.4) gives for

[1] This is based upon the 8,388,607-bit sequence found in Section 5.6 of the ITU-T O.150 standard.

[2] Figure 24, 'AD9271 Octal LNA/VGA/AAF/ADC and Crosspoint Switch Data Sheet (Rev. B)', Analog Devices, May 2009. `http://www.analog.com/static/imported-files/data_sheets/AD9271.pdf`

Analogue supply	Digital supply	Mean	Variance
Linear	Linear	-12.6	2.13
Switching	Linear	-12.3	2.65
Linear	Switching	-12.3	2.60
Switching	Switching	-12.1	2.73

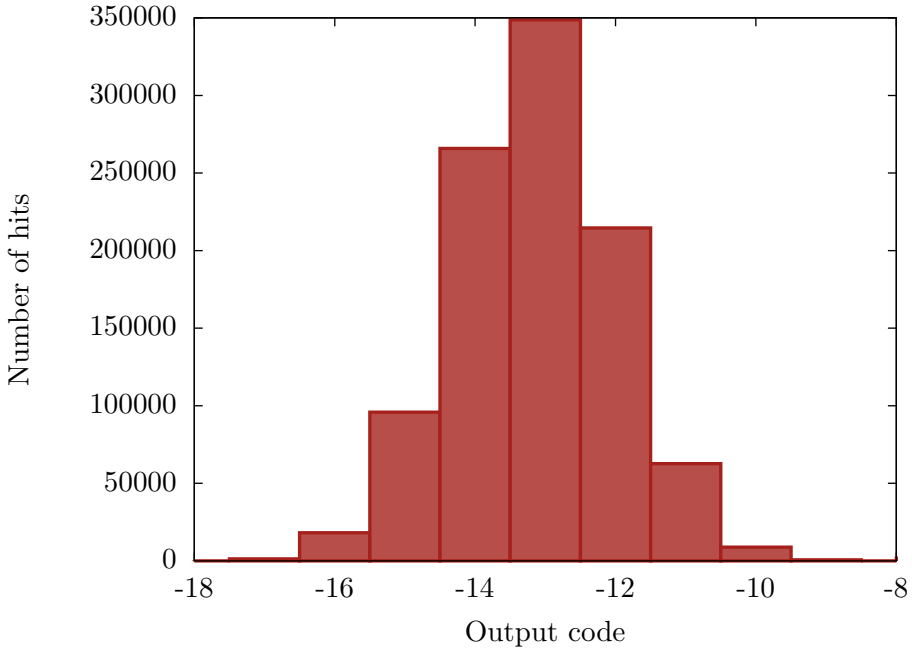
Table 5.1: The effect of using different regulators on the mean and variance of the AD9271 output when its inputs were shorted.

a 12-bit ADC.

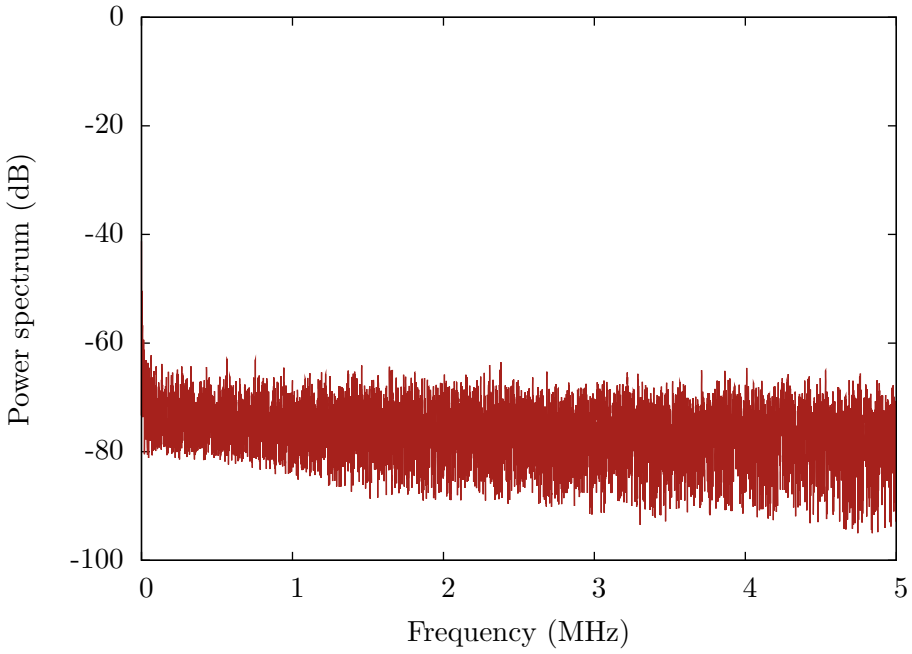
The T/R switch was then added to the circuit to see if it introduced further noise to the system. With the input to the T/R switch short-circuited, around one million samples were collected. The histogram of these samples, given in Figure 5.2a, has a mean of -11.8 and a variance of 3.1. This increase in variance indicates some extra noise in present, and this is confirmed by the PSD of these samples, shown in Figure 5.2b, having a noise floor of -67 dB, approximately 3 dB higher than without the T/R switch.

5.3.2 Power supplies

As mentioned in Section 4.3.2, the 1.8 V for the analogue and digital supplies of the AD9271 can be generated by either a linear or switching regulator. The switching regulator will be more efficient, but may introduce noise at its switching frequency. In the other tests in this chapter, both of the supplies were powered from the linear rectifiers. To evaluate the noise levels from the different regulator combinations, the inputs to the AD9271 were short-circuited and around one million samples collected for each combination (both supplies taken from linear regulators, both taken from switching regulators, and one supply taken from a linear regulator and the other from a switching regulator). The resulting histogram is shown in Figure 5.3, and the corresponding statistics are presented in Table 5.1. These show a change in the mean between supplies, backing up the theory that this offset is caused by a reference error. The switching regulators do result in an increase in the variance and hence the noise in the output. However, it is not a major increase, with PSD analysis showing the largest increase (when both supplies are taken from a switching regulator) to be around 1.5 dB.

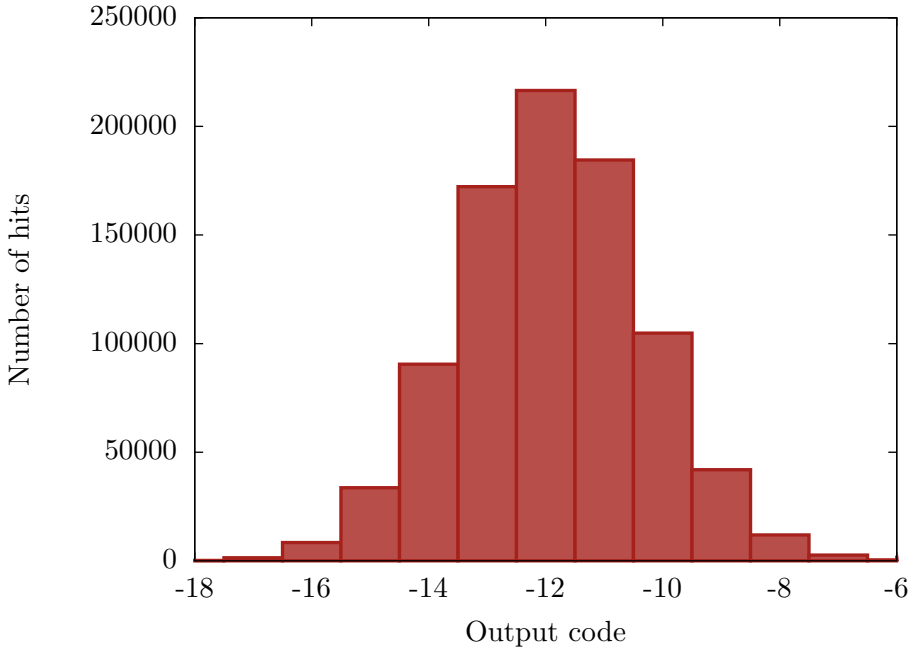


(a) Histogram

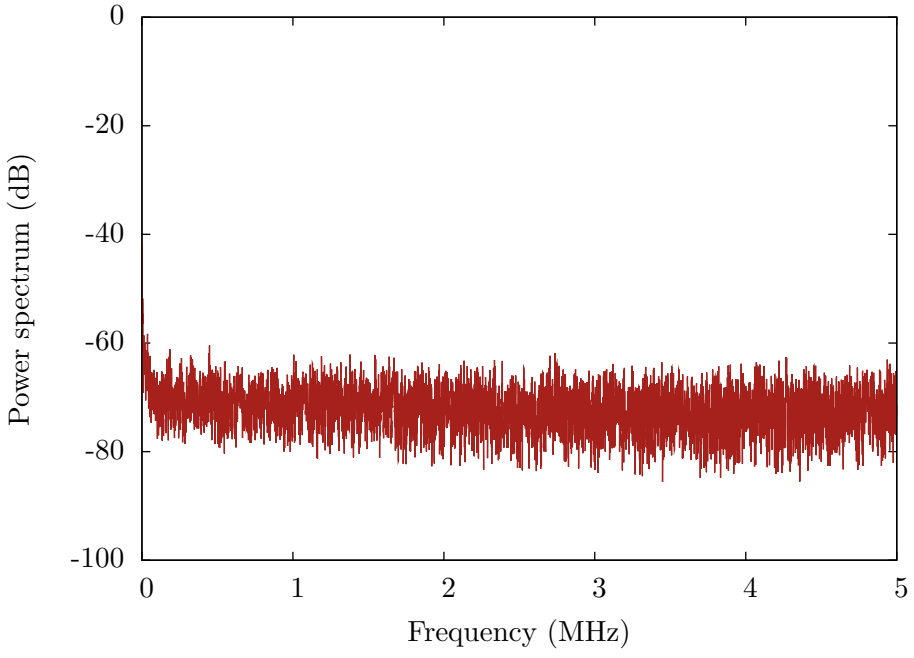


(b) Power spectral density

Figure 5.1: The noise performance of the AD9271 with its inputs short-circuited.



(a) Histogram



(b) Power spectral density

Figure 5.2: The noise performance of the receiver with the inputs to the T/R switch short-circuited.

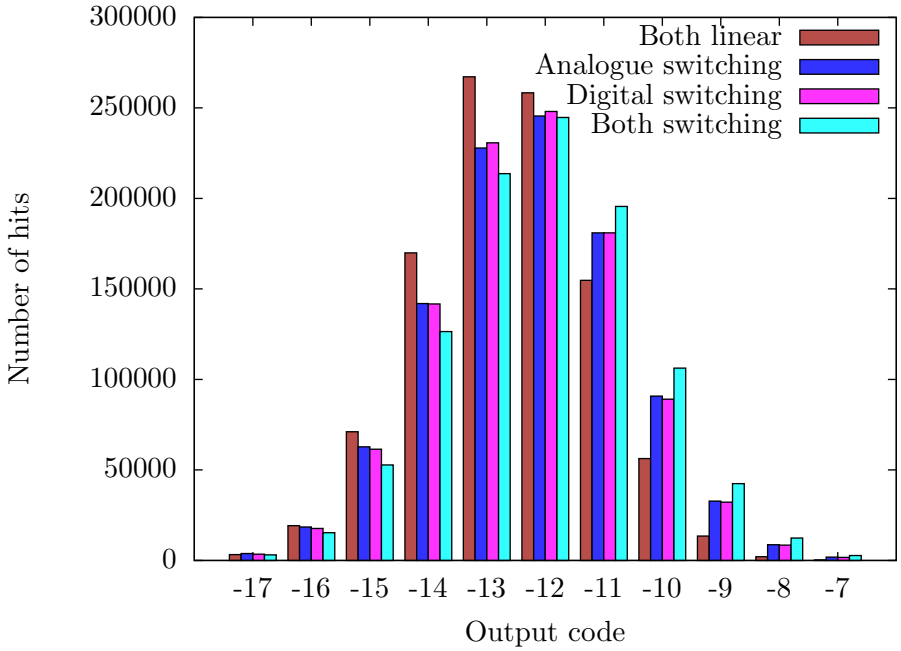
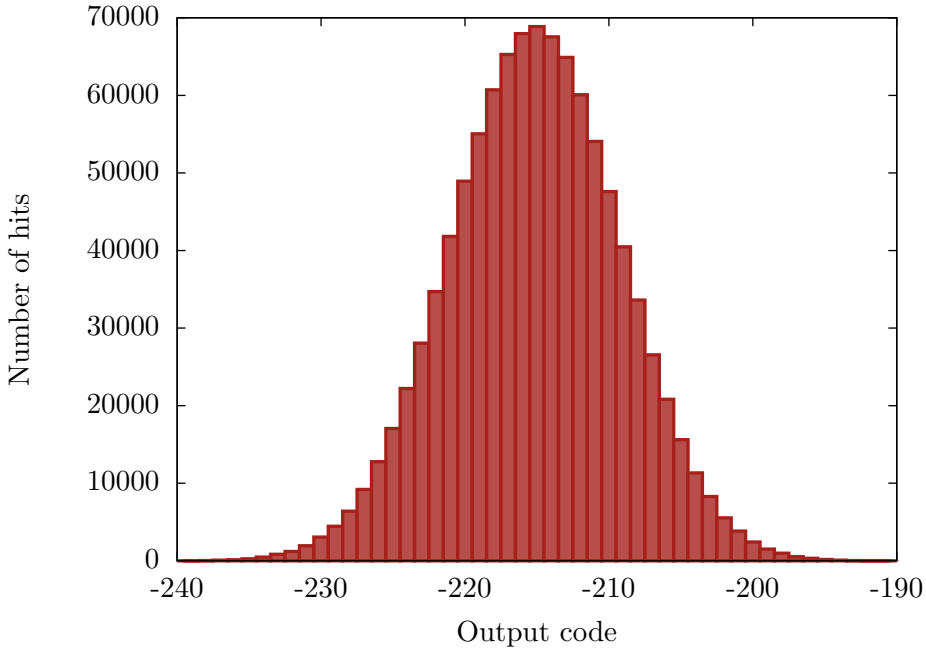


Figure 5.3: A histogram showing the noise performance of the receiver with the different power supply options. The inputs to the AD9271 were short-circuited when collecting this data.

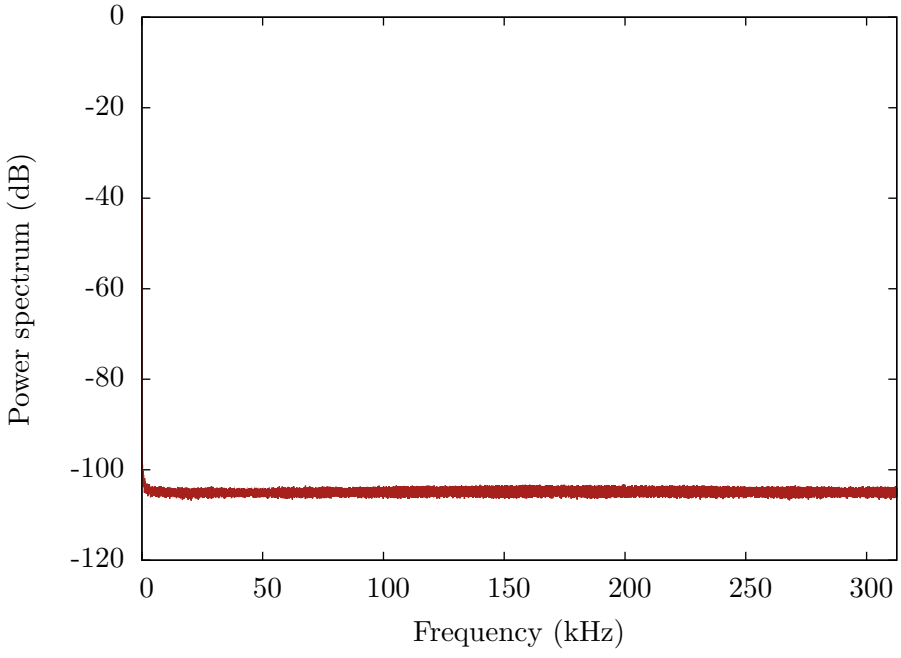
5.3.3 CIC filter

After decimation, a lower noise floor would be expected due to the increase in resolution. To test this, a CIC filter with $R = 16$, $M = 1$ and $N = 1$ was used to decimate the outputs of the AD9271, with the resulting data having a width of sixteen bits. A histogram of just over a million of these outputs is given in Figure 5.4a, and reveals a larger offset (the mean is -215.2) and variance (45.5) than the previous tests due to the increase in the size of the number system. The PSD of these samples was calculated using the same method as previously. Shown in Figure 5.4b, it gives the noise floor of the decimated data as approximately -105 dB, an improvement of around 35 dB on the samples from the AD9271 alone and below the 98 dB that (3.4) predicts for a sixteen bit ADC.

The final noise test performed was to see if the noise introduced by the T/R switch could be reduced by decimation. With the T/R switch inputs short-circuited and the same CIC filter as before decimating the AD9271 output, one million samples were collected. The corresponding histogram, shown in Figure 5.5a, has a mean of -213.0



(a) Histogram



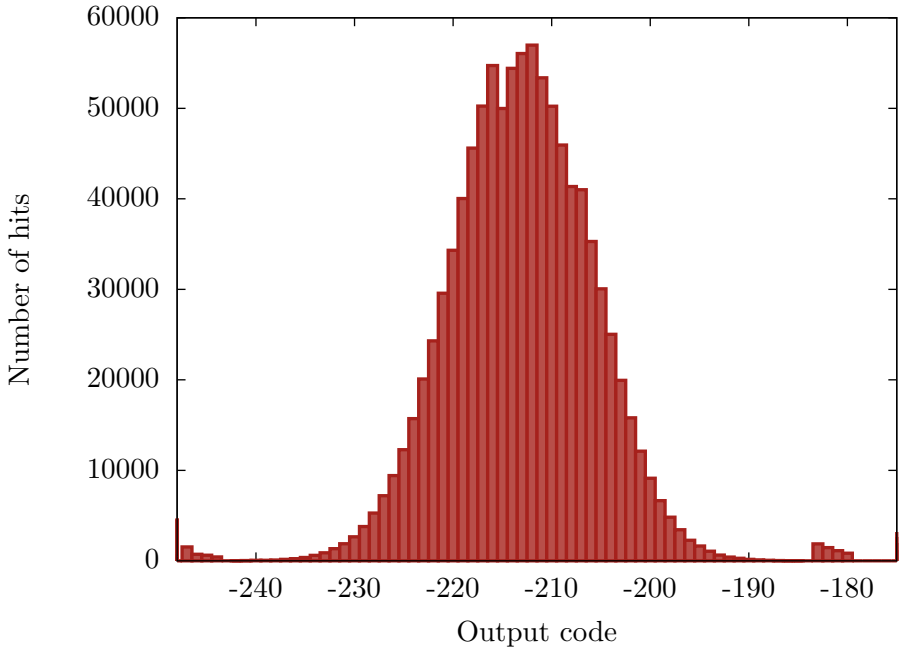
(b) Power spectral density

Figure 5.4: The noise performance of the AD9271 with its inputs short-circuited after decimation with a CIC filter ($R = 16$, $M = 1$, $N = 1$).

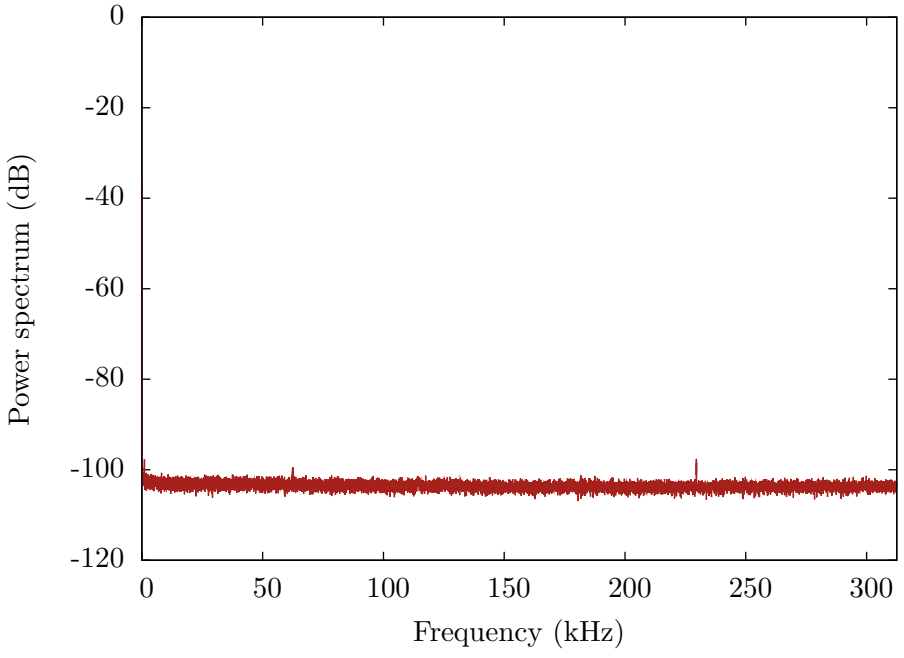
and a variance of 55.4. This increased variance, along with the uneven shape of the histogram and the presence of outliers, suggests there is more noise present than in the previous test. As shown in Figure 5.5b, the PSD gives the noise floor as -102 dB. This is a 3 dB increase compared to the level of noise from the previous test, but approximately 35 dB lower than without decimation.

5.4 System testing

Finally, the ability of the system to transmit and receive a ping was tested. The prototype PCB is reliant on external power supplies to provide the 12 V and 200 V needed for the transmitter along with the -5 V for the T/R switch. This meant the test had to be performed in the laboratory with the transducer placed in a bucket of water. A ping was transmitted with the received signal shown in Figure 5.6. After the transmission has ceased, the resonant nature of the transducer means that ringing continues. In this case, it takes about 1 ms for the ringing to die away to a point where the T/R switch is no longer clipping the signal. The direct path to the far edge of the bucket and back was around 1 m, which takes around 650 μs for the signal to travel. As a result, the direct echo is swamped by the ringing. However, multipath echoes — i.e, where the signal has bounced off multiple surfaces before returning, such as the bottom of the bucket followed by the side — are visible after the ringing has died away. For example, the echo at 2.5 ms corresponds to a total distance travelled of around 3.75 m. These results confirm that the design is capable of transmitting a ping and receiving the echoes.



(a) Histogram



(b) Power spectral density

Figure 5.5: The noise performance of the receiver with the inputs to the T/R switch short-circuited after decimation with a CIC filter ($R = 16$, $M = 1$, $N = 1$).

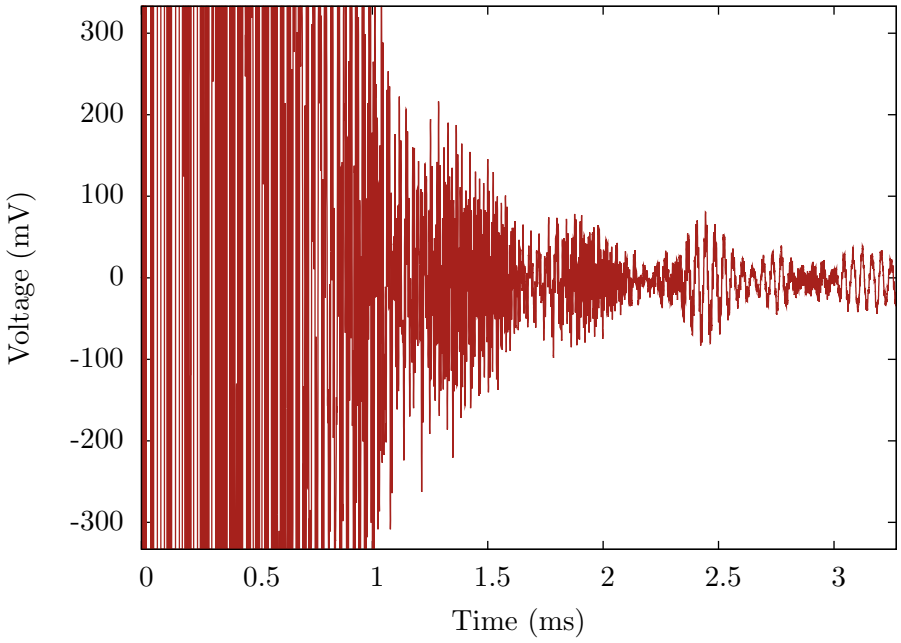


Figure 5.6: The transmitted signal and received echo with the transducer placed in a bucket of water. Due to the short range, the direct echo is swamped by the ringing of the transducer. The echoes visible later in the signal have reflected off multiple surfaces. For instance, the echo at 2.5 ms has travelled around 3.75m.

CHAPTER 6

CONCLUSION

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Sir Winston Churchill
On the victory at El Alamein, 10 November 1942

CHAPTER 6

CONCLUSION

A new multi-channel front-end system has been developed for the SAS operated by the Acoustics Research Group at the University of Canterbury. The transmitter uses a class D switching amplifier to generate the output waveform, with the resonant nature of the transducer used to filter the switching harmonics. The highly integrated AD9271 amplifier and ADC chip was chosen to condition and sample the received signal since it is able to handle eight channels simultaneously. As the transducers are used for both transmitting and receiving in this design, a T/R switch is used to prevent the high-voltage output signal from damaging the receiver circuitry. Due to the high output data rate of the AD9271, an FPGA is required to decimate the sampled signal, with multiplier-free CIC filters being used to minimise the computational effort required for the decimation.

A prototype PCB has been created to evaluate the performance of the design. The class D amplifiers operated as designed as did the T/R switch. The drain-to-source resistance of the MOSFETs when the amplifier was turned off formed a voltage divider and resulted in a 200 mV DC offset being added to the returning signal, potentially causing the T/R switch to block some received echoes. When enabled, the blocking capacitor provided in the design prevented this offset from propagating to the transducer without noticeably affecting the output of the amplifier. The LC filter included in the transmitter circuit did not operate as intended, but instead provided a low-impedance path to ground which resulted in a large current being drawn and destroying the MOSFETs.

Firmwave has been developed to control the FPGA and evaluate the performance of the AD9271. Noise testing showed that the noise generated by the AD9271 itself

conformed with its specifications, and that the extra resolution gained from decimating the sampled signal lead to a significant reduction in the noise floor of the signal. The T/R switch adds around 3 dB of noise, with the noise level of the T/R switch and receiver after decimation being measured as -102 dB. Further testing showed that the design was capable of transmitting and receiving a pulse of acoustic energy.

These tests showed that the design presented in this thesis is capable of transmitting and receiving the ping used by the sonar. When coupled with the back-end system currently under design, it will be able to replace the existing electronics in the KiwiSAS-IVsystem.

6.1 Future work

Since this design was started, Analog Devices have released two companion chips to the AD9271, namely the AD9272 and AD9273. These have the same basic features — eight channels, integrated LNA and VGA, twelve bit ADC — but with differing sampling speeds, maximum possible gain, and noise levels. It would be worth investigating these other chips for future implementations of the front-end.

The VHDL developed in the course of this thesis should be sufficiently scalable to control the full system. However, further development is required to create a suitable interface for communications with the back-end system. As development of the back-end system has just started, the parameters for this interface have not been decided yet. The current intention is for the back-end to be based around a powerful microprocessor, such as the ARM9. It is envisaged that two serial links, based upon SPI or a similar protocol, will be used for communications, one used to control the front-end and the other dedicated to returning the decimated data. The latter will require firmware to collate the data from all the channels and serialise it, while the control interface will require a receiver to interpret the incoming commands and apply them to the appropriate modules.

There is scope for the firmware to be developed to perform further processing on the received signals before they are transferred to the back-end system. For example, one of the first steps performed when reconstructing an image from the signals is to

split the two bands used in the signal (20–40 kHz and 90–110 kHz) and shift them to baseband. If this is performed in the FPGA, a further reduction in the sampling rate would be possible, thus reducing the storage requirements.

APPENDIX A

SCHEMATICS

The schematics for the prototype board are reproduced in this appendix. They were laid out in Altium Designer in a hierarchical fashion i.e., the 'low-level' schematics such as the circuitry for a single channel are included in higher level schematics to build up the design.

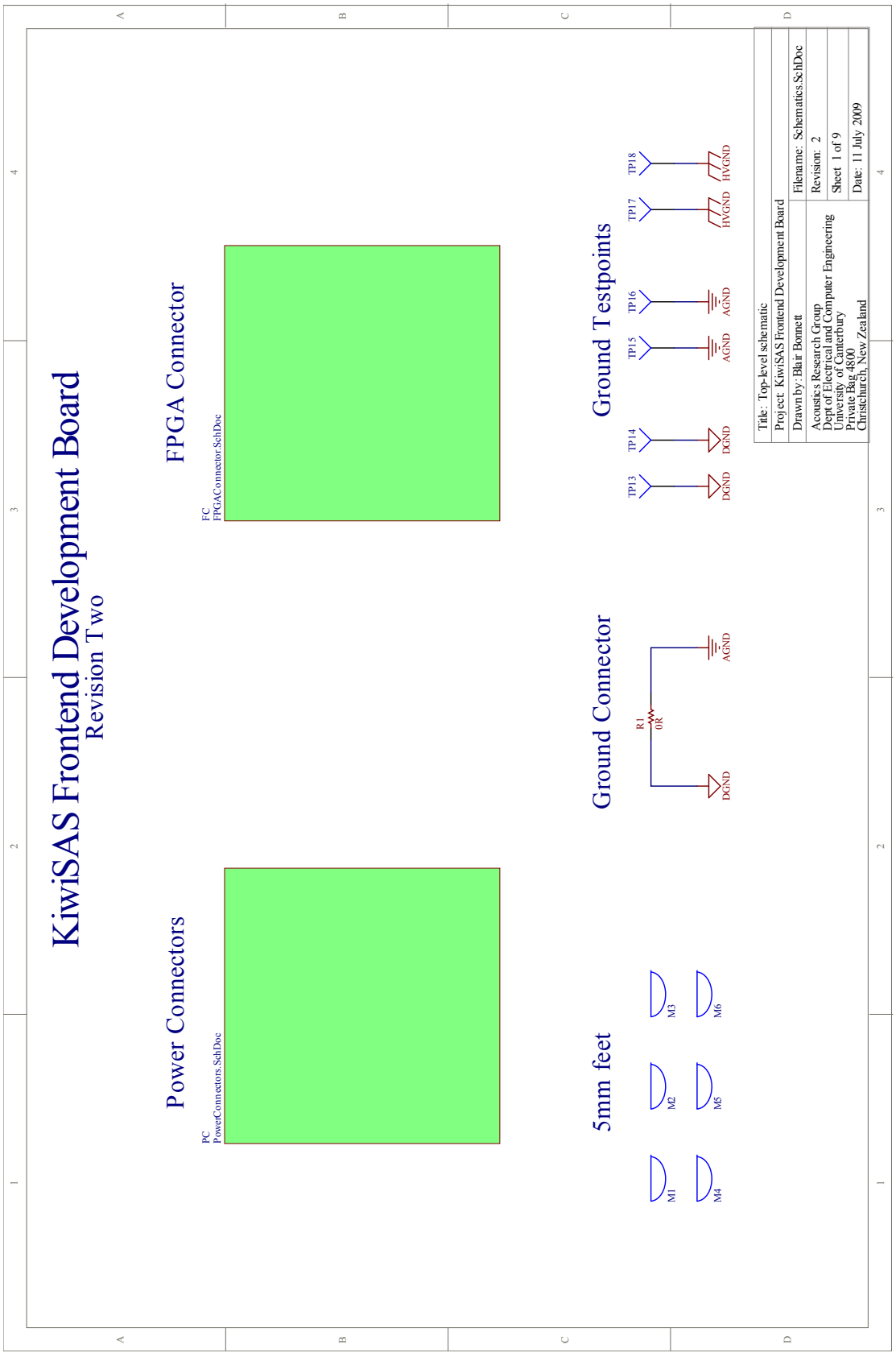


Figure A.1: The top-level schematic.

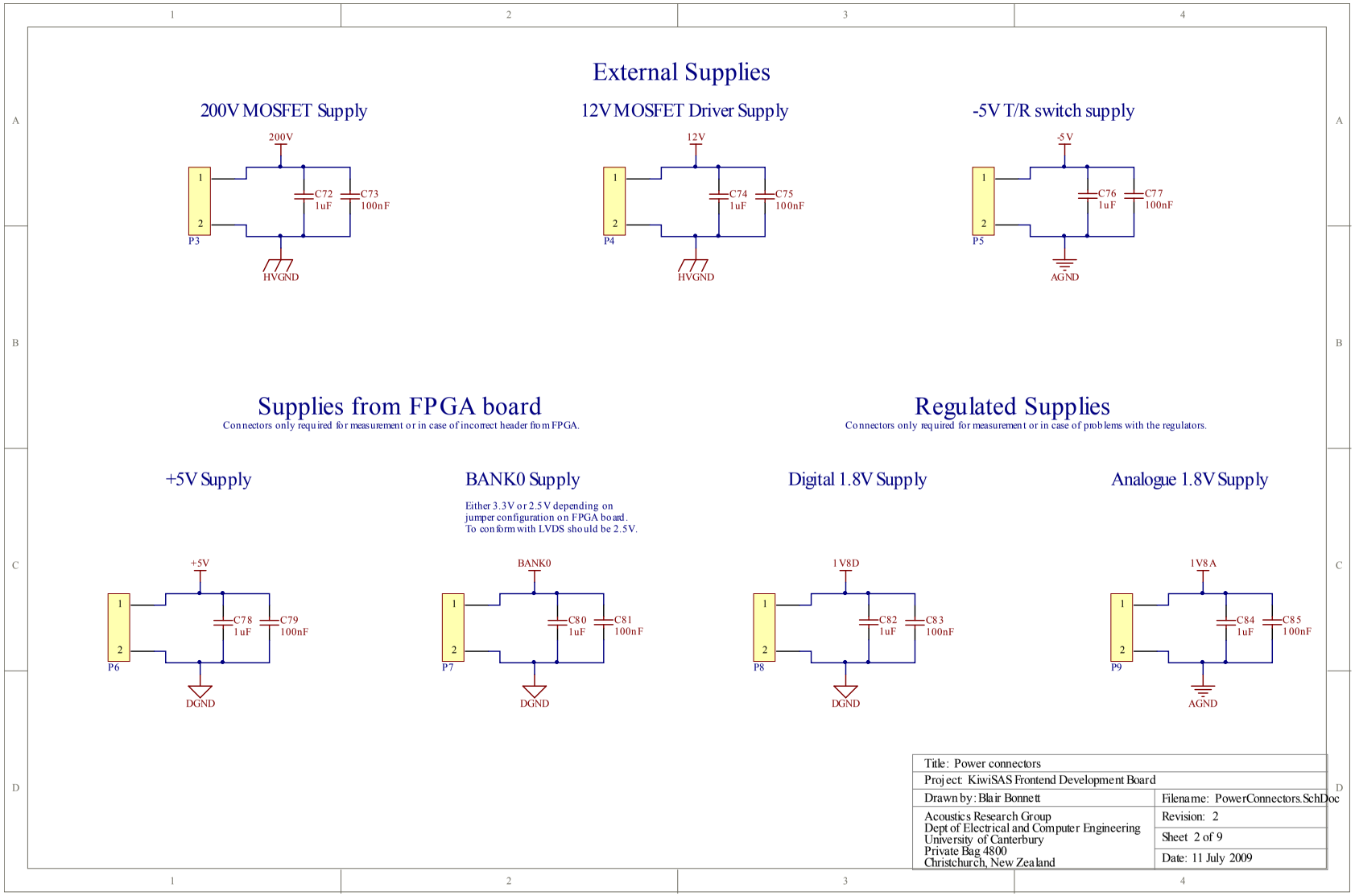


Figure A.2: The power connectors schematic.

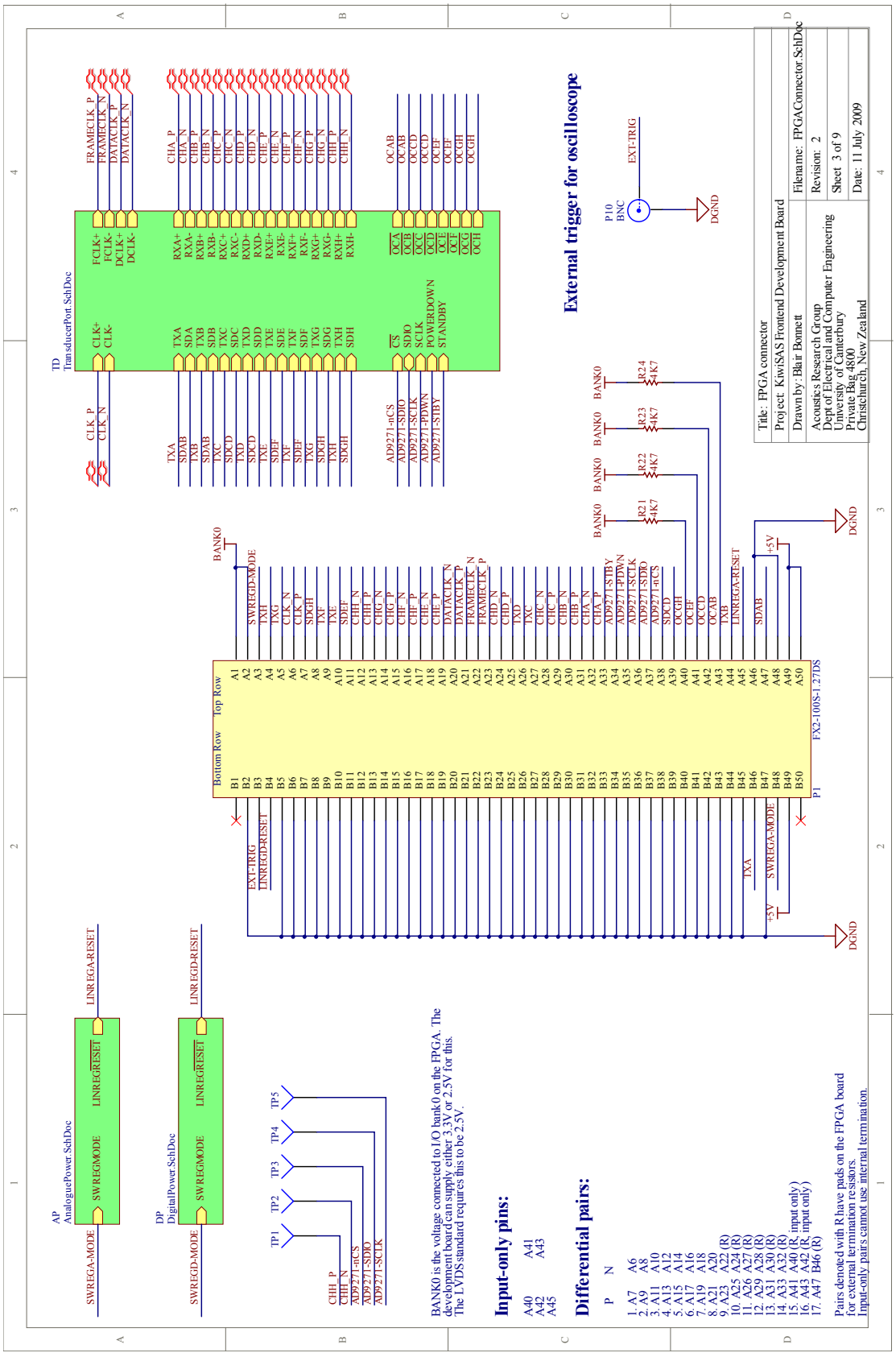


Figure A.3: The FPGA connector schematic.

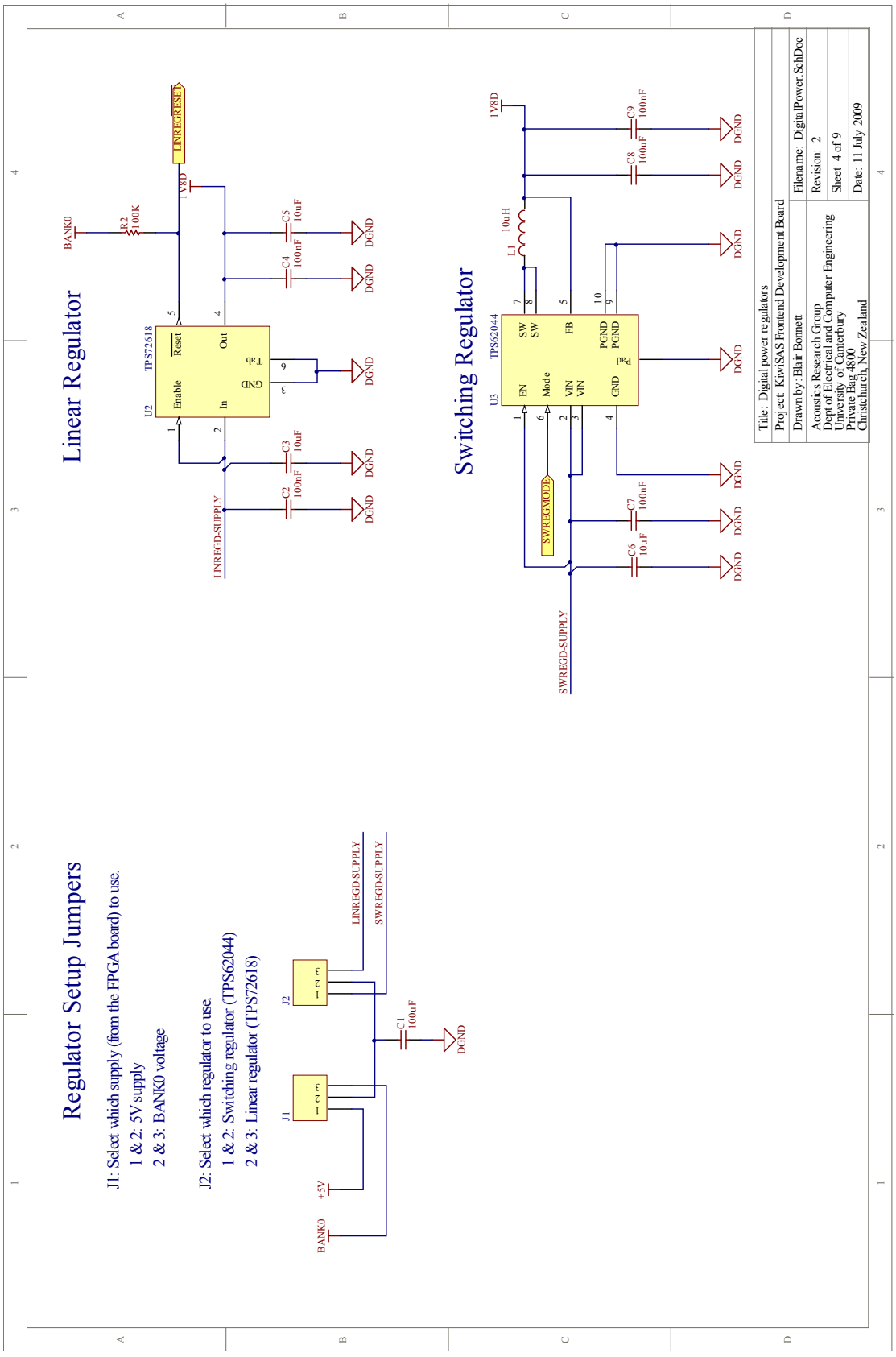


Figure A.4: The digital power regulators schematic.

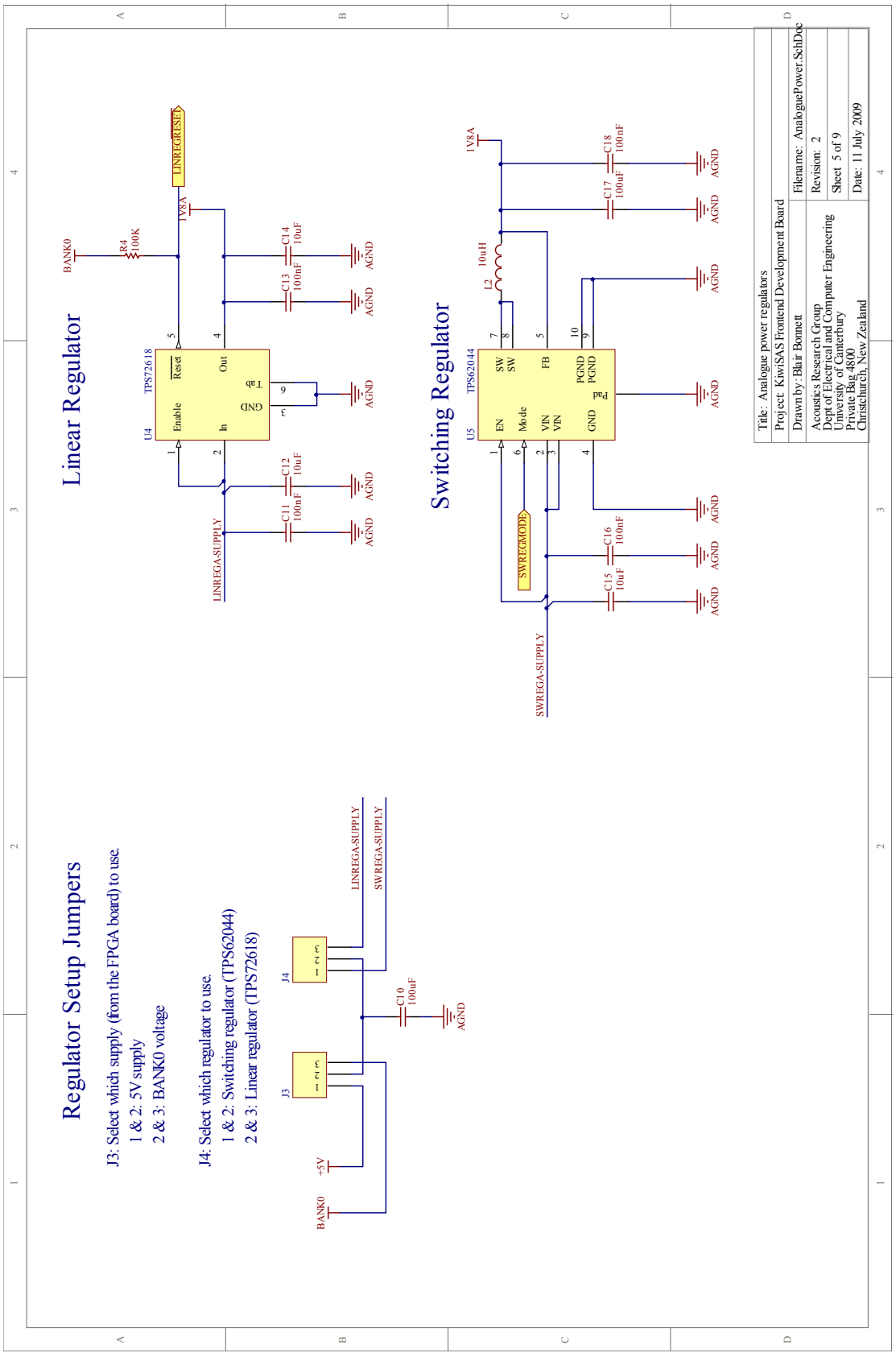


Figure A.5: The analogue power regulators schematic.

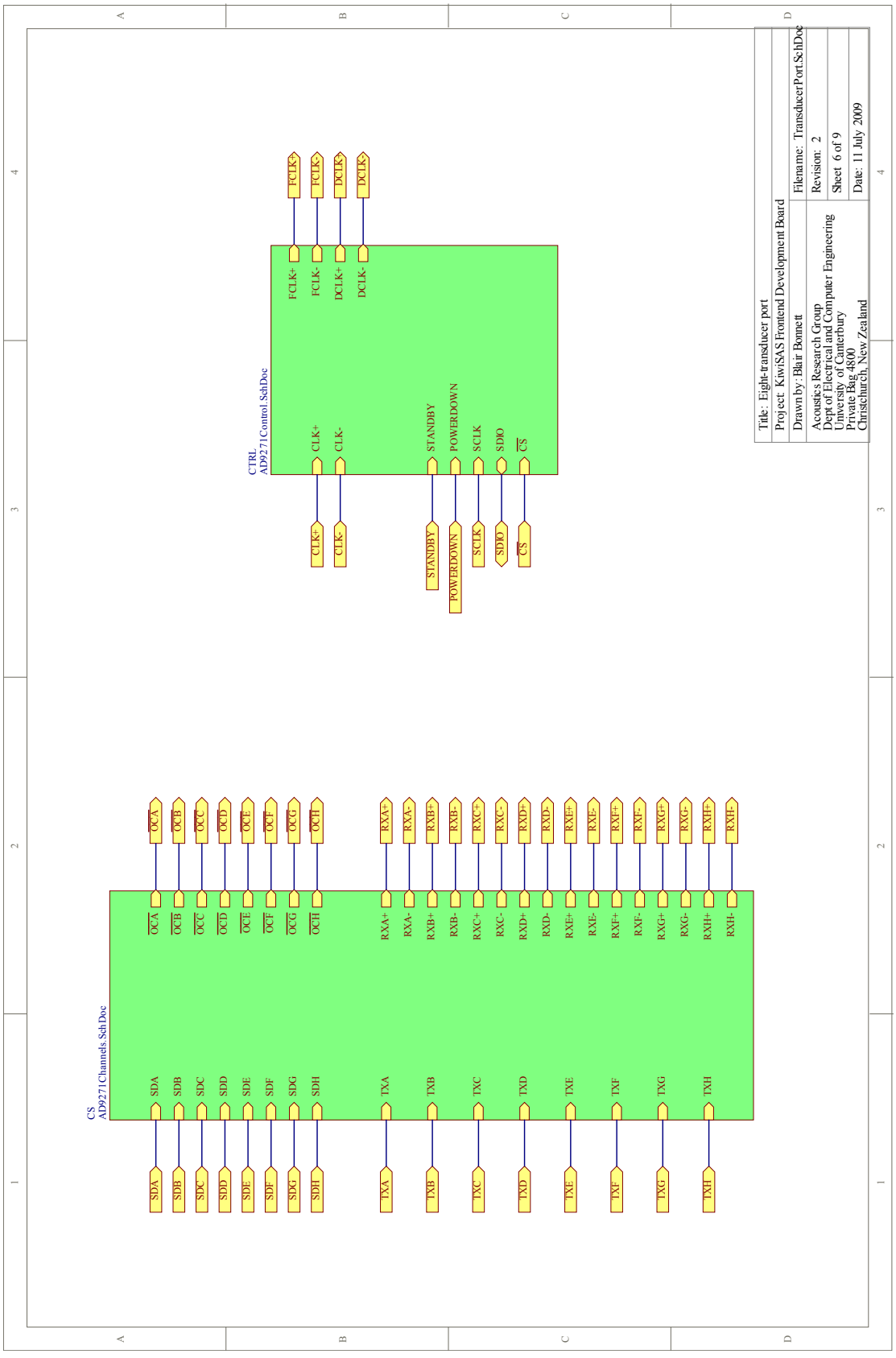


Figure A.6: The transducer port schematic.

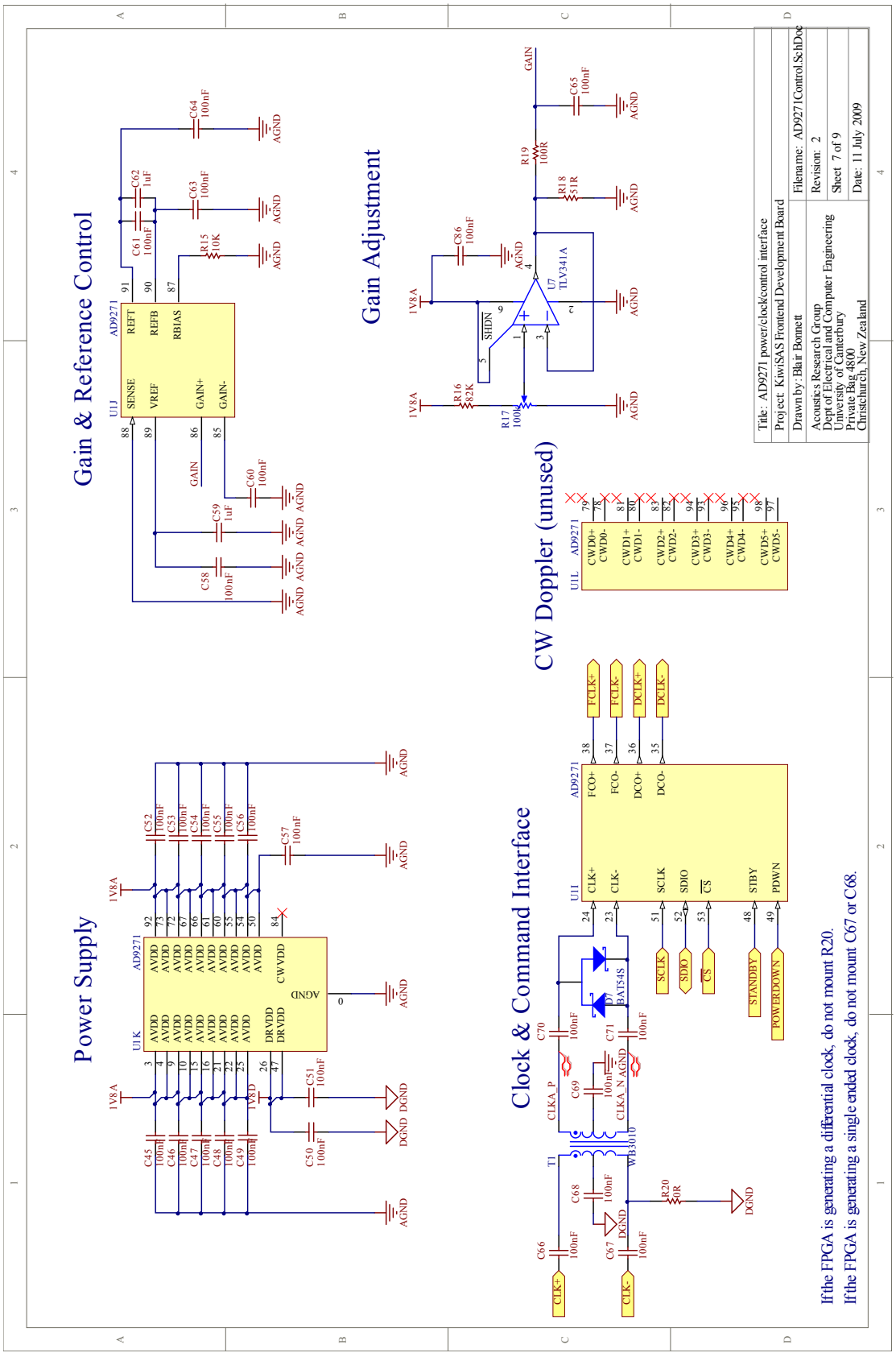


Figure A.7: The AD9271 power and control schematic.

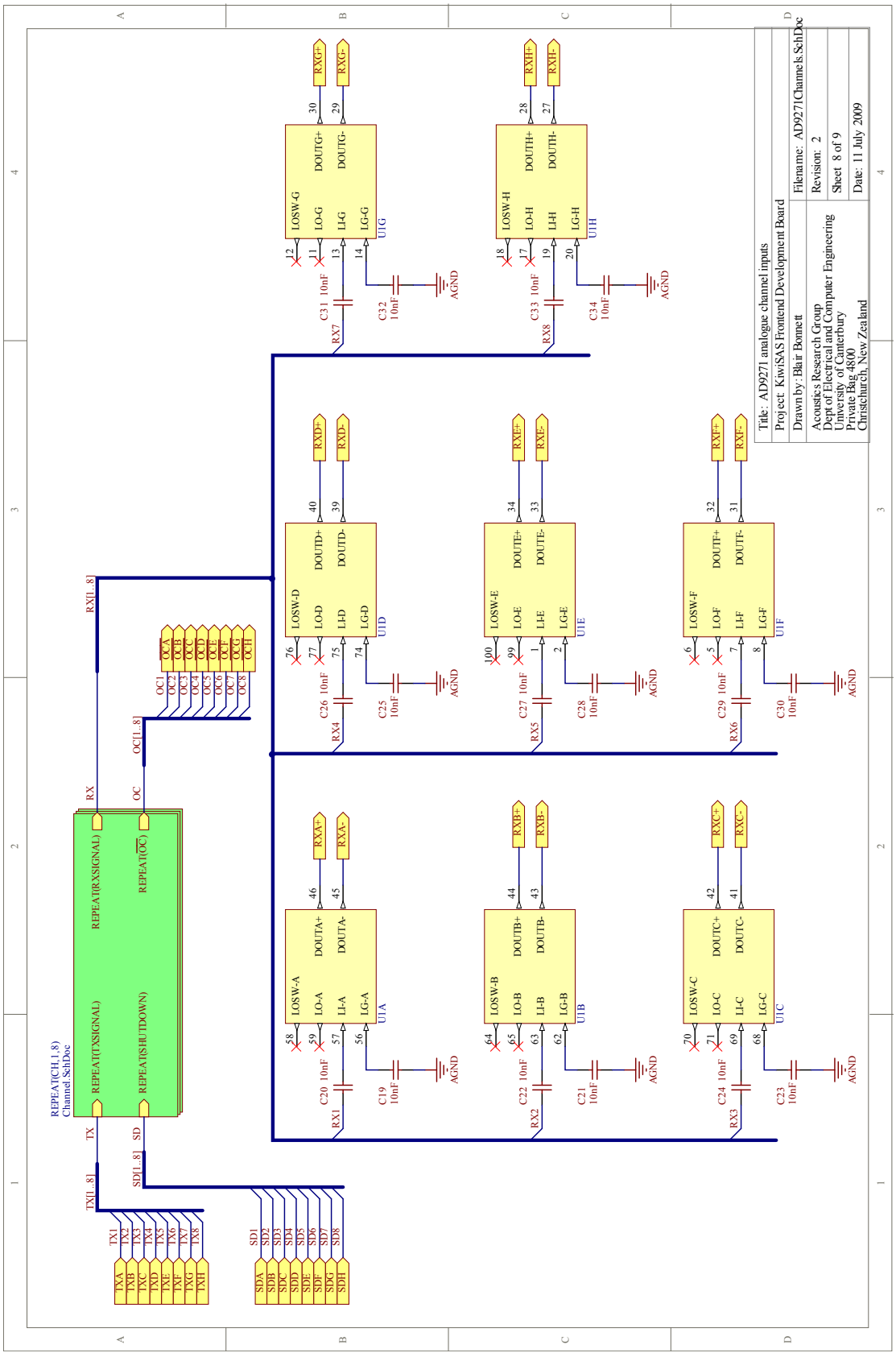


Figure A.8: The AD9271 analogue channel inputs schematic.

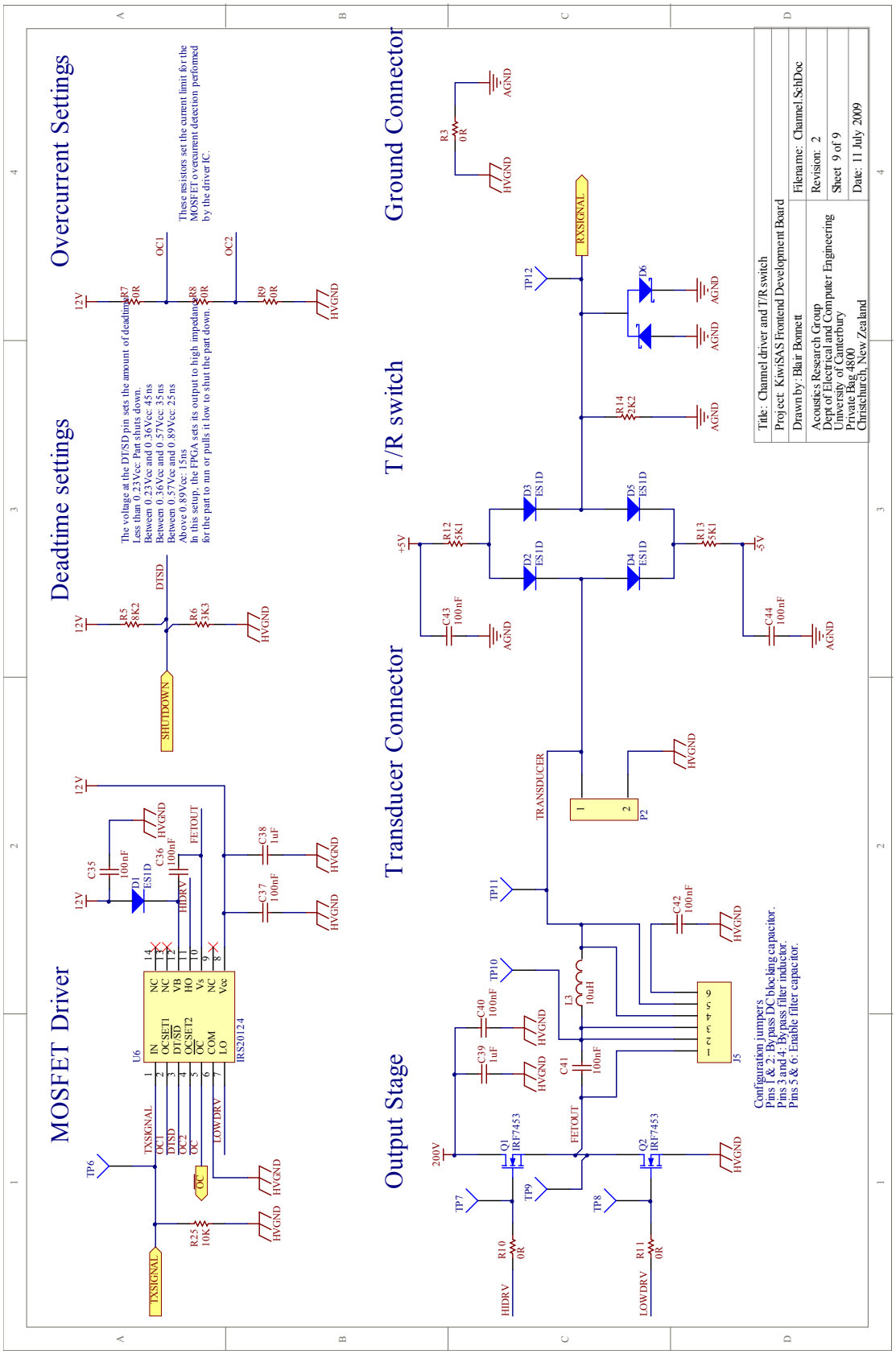


Figure A.9: The channel driver and T/R switch schematic.

APPENDIX B

PCB LAYOUT

This appendix shows the four layers used in the prototype PCB. The layout of the PCB is detailed in Section 4.3.2.

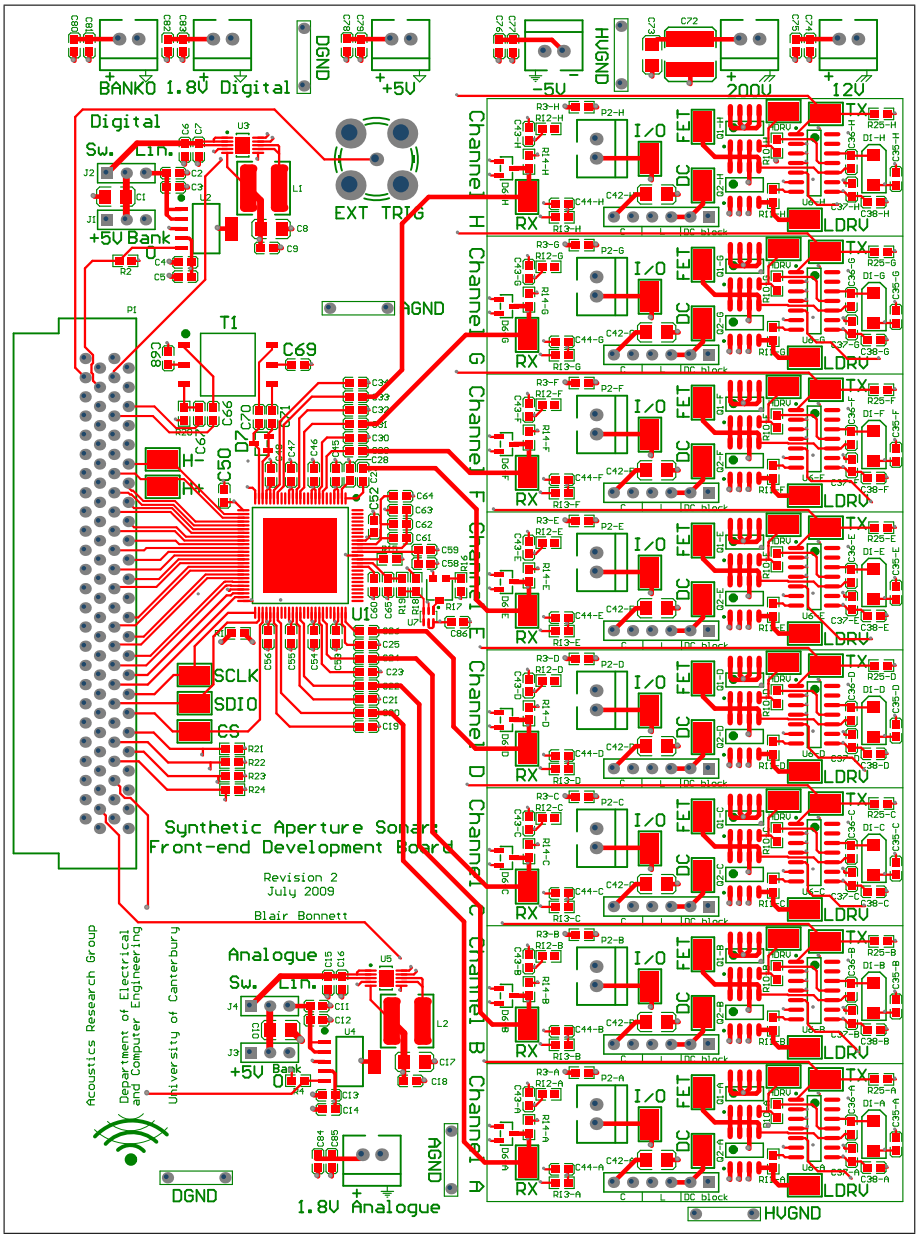


Figure B.1: The top layer of the PCB. The red is the copper and the green is the silkscreen.

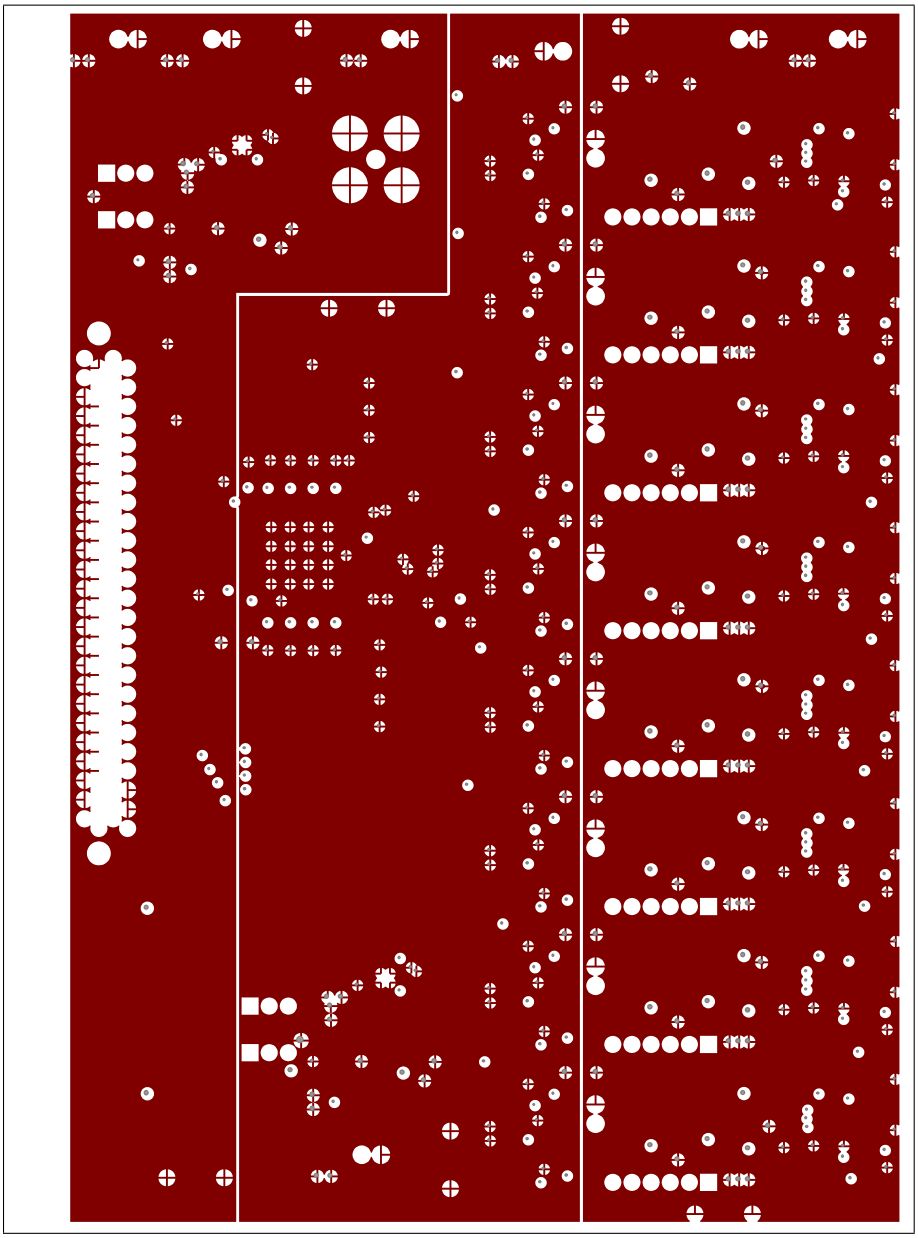


Figure B.2: The PCB ground planes. The left-hand plane is the digital ground, the middle plane is analogue ground and the right-hand plane is high voltage ground.

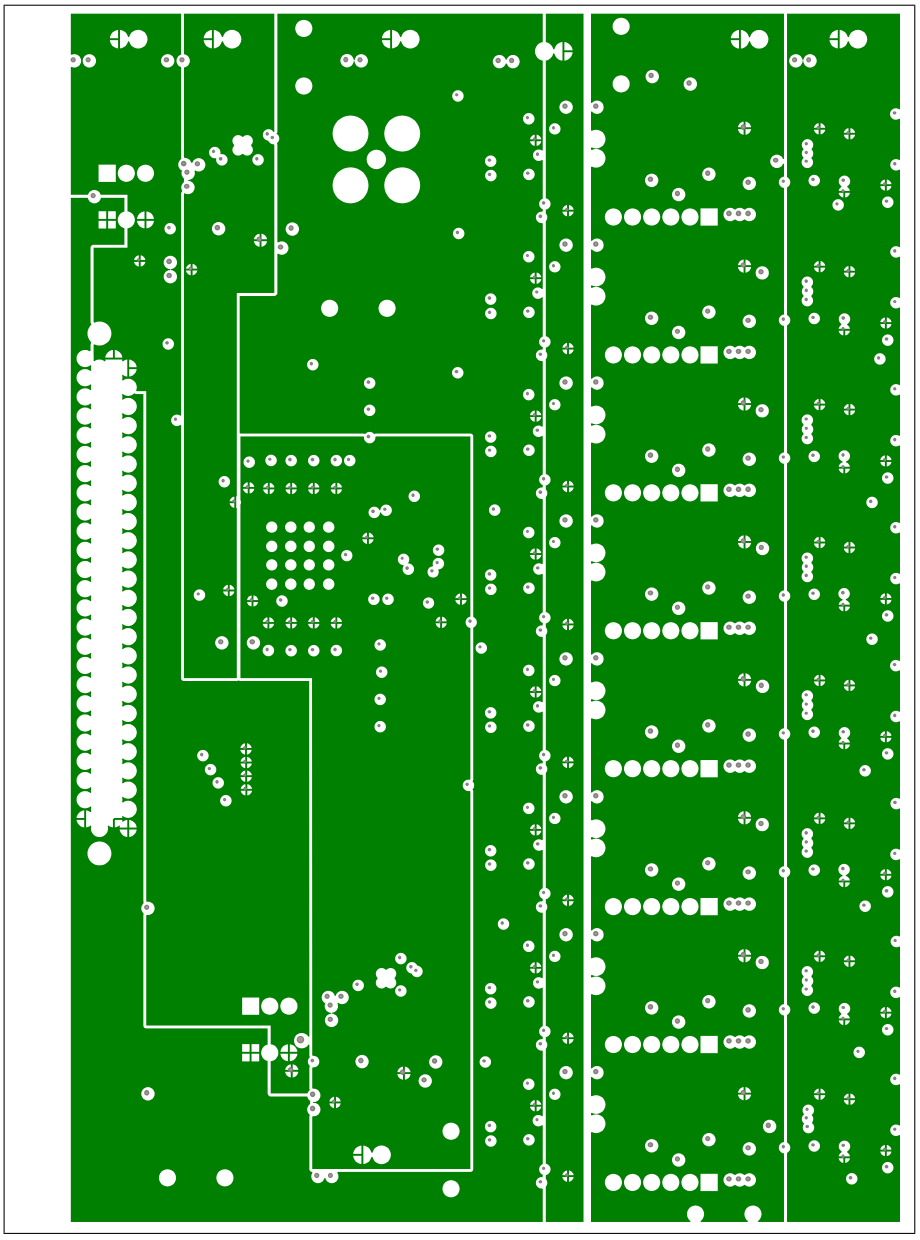


Figure B.3: The PCB power planes.

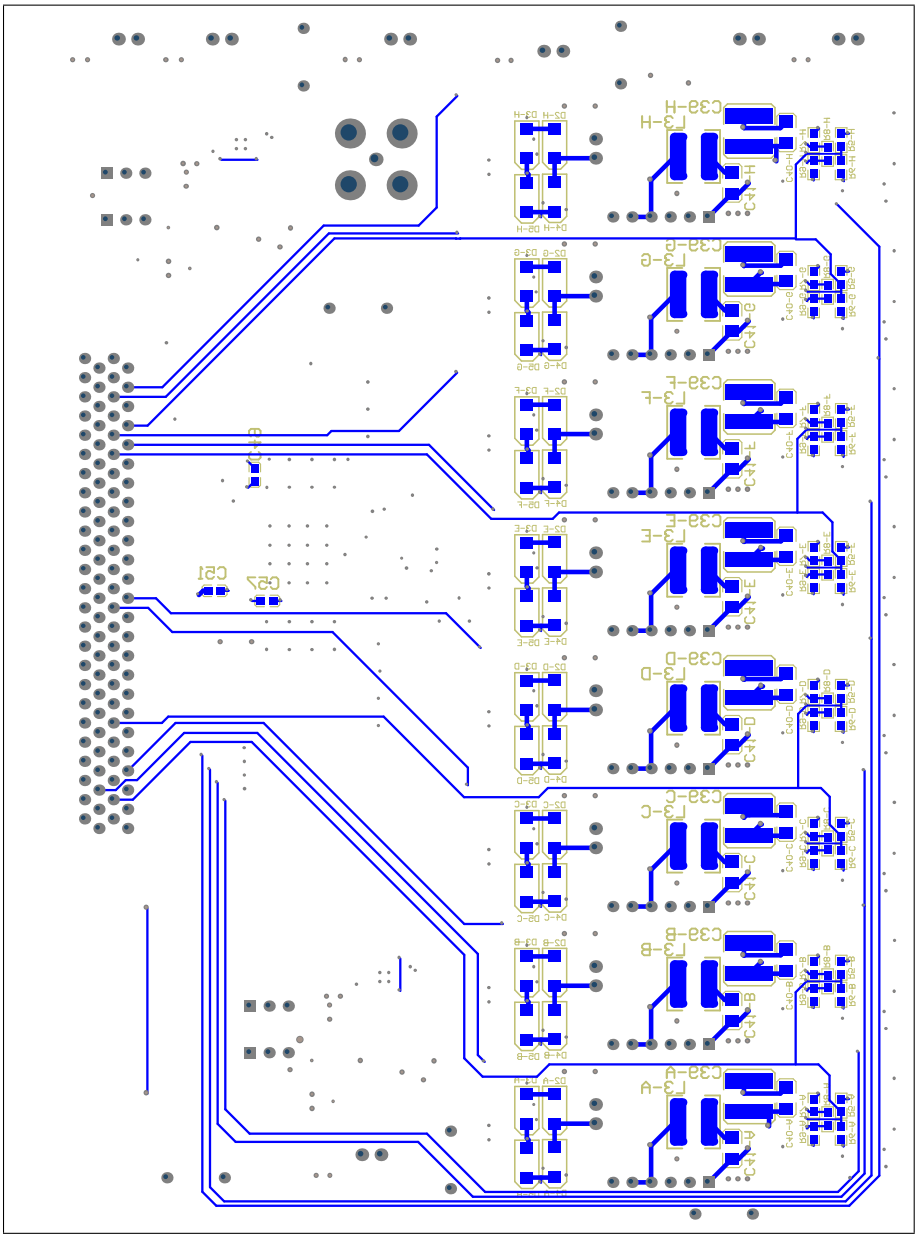


Figure B.4: The bottom layer of the PCB. The blue is the copper and the beige is the silkscreen.

APPENDIX C

VHDL CODE

This appendix contains the VHDL code for the firmware developed in this thesis. The design of these modules is described in Section 5.1.

C.1 Debouncer

```
-- debounce: Debounce a signal using a low-pass filter and Schmitt
--           trigger.
--
-- This VHDL component is designed to debounce a noisy signal into a
-- clean signal synchronised to a clock. It does so by implementing
-- a simple IIR low-pass filter followed by a Schmitt trigger.
--
-- Inputs:
--   * sampleclock: Clock signal controlling how often the input
--                  is sampled.
--   * synchclock: Clock signal to which the outputs are synchronised.
--   * input: The signal to be debounced.
--
-- Outputs:
--   * current: The state of the debounced signal. Updated on the
--              rising edge of synchclock.
--   * prev: The state of the debounced signal at the previous
--           rising edge of synchclock. Can be used to detect edges.
--
-- Version: 1.0
-- Last updated: 26 October 2009
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity debounce is
   port(
      sampleclock : in std_logic;
      synchclock : in std_logic;
      input : in std_logic;
      current : out std_logic := '0';
      previous : out std_logic := '0'
```

```
   );
end debounce;

architecture behavioral of debounce is
   signal value : unsigned(7 downto 0) := (others => '0');
   signal cur : std_logic := '0';
   signal prev : std_logic := '0';

begin

   -- Perform the low pass filtering.
   -- The difference equation is y[n+1] = 0.75y[n] + x[n].
   -- A zero input is mapped to 0, and a one input to 63.
   -- We do the multiplication by 0.75 by subtracting 0.25 of the value.
   -- A division by 4 is simply two right shifts.
   dofilter : process(sampleclock) is
   variable temp : unsigned(7 downto 0);
   begin
      if sampleclock'event and sampleclock = '1' then
         if input = '1' then
            value <= value - value(7 downto 2) + to_unsigned(63, 8);
         else
            value <= value - value(7 downto 2);
         end if;
      end if;
   end process dofilter;

   -- Implement a Schmitt trigger, synchronising outputs to synchclock
   doschmitt : process(synchclock) is
   begin
      if synchclock'event and synchclock = '1' then
         prev <= cur;

         -- Schmitt trigger is currently low, check if we have crossed
         -- the threshold.
         if cur = '0' then
            if value > 240 then
               cur <= '1';
            else
               cur <= '0';
            end if;

         -- Schmitt trigger is currently high, check if we have crossed
         -- the threshold.
         else
            if value < 15 then
               cur <= '0';
            else
               cur <= '1';
            end if;
         end if;
      end if;
   end process doschmitt;

   -- Output signals
   current <= cur;
   previous <= prev;
end behavioral;
```

C.2 RS232 data collection

```
-- rs232tx: RS232 transmitter module with inbuilt buffer.
--
-- This VHDL component implements a RS232 transmitter which can be
-- configured via generic parameters. The component includes block
-- RAM which is used to store the incoming data (i.e., a set of data
-- can be clocked in at a much higher rate than it is transmitted).
-- It can also handle sending data over multiple transmissions so
-- that the incoming data can be wider than the number of RS232 data
-- bits.
--
-- Generic parameters:
--   * SYSTEM_CLOCK_SPEED: The speed of the clock that is used to
--                         control the transmissions.
--   * BAUD_RATE: The desired RS232 baud rate.
--   * DATA_BITS: The number of RS232 data bits to be used.
--   * PARITY: Whether to use parity. 1 means odd parity, 2 means even
--             parity, and anything else means don't use parity.
--   * STOP_BITS: The number of stop bits to send. Must be 1.0 or
--                greater otherwise transmission will not work.
--   * BUFFER_WIDTH: The number of bits for each entry in the buffer.
--   * BUFFER_ENTRIES: The number of entries to store in the buffer.
--   * PADDING_MODE - If the input data does not fit evenly into one or
--                    more RS232 blocks, the MSBs of the final block
--                    will be padded. If this parameter is 0, they will
--                    be padded with zeroes; if it is 1, they will be
--                    padded with ones and if it is 2, the MSB of the
--                    actual data will be used (i.e., it will be sign
--                    extended). Any other number will result in the
--                    data being padded with zeroes.
--
-- Inputs:
--   * dataclk: The data is loaded into the buffer on the rising edge
--              of this clock.
--   * dataen: Enables the buffer so data can be stored.
--   * data: The data to transmit.
--   * sysclk: The main system clock.
--
-- Outputs:
--   * bufferempty: If the buffer is currently empty.
--   * bufferfull: If the buffer is currently full.
--   * tx: The RS-232 output to be transmitted.
--
-- Version: 1.1
-- Last updated: 2 November 2009
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity rs232tx is
   generic(
      SYSTEM_CLOCK_SPEED : positive := 50000000;
```

```
      BAUD_RATE          : positive :=   115200;
      DATA_BITS          : positive :=        8;
      PARITY             :  natural :=        0;
      STOP_BITS          :     real :=      1.0;
      PADDING_MODE       :  natural :=        2;
      BUFFER_WIDTH       : positive :=        8;
      BUFFER_ENTRIES     : positive :=     1024
   );

   port(
      dataclk     : in std_logic;
      dataen      : in std_logic;
      data        : in std_logic_vector(BUFFER_WIDTH - 1 downto 0);
      sysclk      : in std_logic;
      bufferempty : out std_logic;
      bufferfull  : out std_logic;
      tx          : out std_logic
   );
end rs232tx;

architecture behavioral of rs232tx is
   -- Data type
   subtype data_t is std_logic_vector(BUFFER_WIDTH - 1 downto 0);

   -- Buffer
   constant BUFFER_ADDR_WIDTH : natural :=
        ...natural(ceil(log2(real(BUFFER_ENTRIES))));
   constant BUFFER_MAX_ENTRY : natural := (2 ** BUFFER_ADDR_WIDTH) - 1;
   type memory_t is array(natural range <>) of data_t;
   signal memory : memory_t(BUFFER_MAX_ENTRY downto 0) := (others =>
        ...(others => '1'));
   signal memoryOutput : data_t := (others => '0');

   -- Buffer address lines
   subtype addr_t is unsigned(BUFFER_ADDR_WIDTH - 1 downto 0);
   signal readAddr : addr_t := (others => '0');
   signal writeAddr : addr_t := (others => '0');

   -- Internal status signals
   signal isbufferempty : boolean := true;
   signal bufferemptyprev : boolean := true;
   signal isbufferfull : boolean := false;

   -- Clock divider ratio
   constant CLOCK_DIVIDER_RATIO_NORMAL : natural := (SYSTEM_CLOCK_SPEED
        .../ BAUD_RATE) - 1;
   constant CLOCK_DIVIDER_RATIO_STOP   : natural :=
        ...(natural(real(SYSTEM_CLOCK_SPEED) * STOP_BITS) / BAUD_RATE)
        ...- 1;

   -- Signal to perform clock division
   subtype clockdiv_t is natural range 0 to CLOCK_DIVIDER_RATIO_STOP +
        ...1;
   signal clockdiv : clockdiv_t := 0;

   -- State machine to handle transmission
   type sendstate_t is (WAITING, SEND_START, SEND_DATA, SEND_PARITY,
        ...SEND_STOP);
   signal sendstate : sendstate_t := WAITING;
```

```
    -- Current byte being transmitted
    signal currentData : data_t := (others => '0');

    -- Position within current byte
    subtype bitpos_t is natural range 0 to DATA_BITS - 1;
    signal bitpos : bitpos_t := 0;

    -- How many 'sends' we have to perform to transmit a complete buffer
         ...entry
    constant SENDS_PER_ENTRY : natural :=
         ...natural(ceil(real(BUFFER_WIDTH) / real(DATA_BITS)));
    subtype whichsend_t is natural range 0 to SENDS_PER_ENTRY - 1;
    signal whichsend : whichsend_t := 0;

    -- Parity bit; synthesizer *should* optimise away if not used
    constant PARITY_USED : boolean := (PARITY = 1 or PARITY = 2);
    signal paritybit : std_logic := '0';

begin

    -- Asynchronously generate and output buffer signals
    isbufferempty <= (readAddr = writeAddr);
    bufferempty <= '1' when isbufferempty else '0';
    isbufferfull <= ((writeAddr + 1) = readAddr);
    bufferfull <= '1' when isbufferfull else '0';

    -- Store incoming data if there is room
    storedata : process(dataclk) is
    begin
       if dataclk'event and dataclk = '1' then
          if dataen = '1' and isbufferfull = false then
             memory(to_integer(writeAddr)) <= data;
             writeAddr <= writeAddr + 1;
          end if;
       end if;
    end process storedata;

    -- Transmit any data in the buffer
    senddata : process(sysclk) is
    begin
       if sysclk'event and sysclk = '1' then
          clockdiv <= clockdiv + 1;

          -- Latch the previous buffer empty signal to use
          -- Without this, sometimes the bufferempty goes false before
          -- the memory output captures the first value and so the
          -- first output is wrong.
          bufferemptyprev <= isbufferempty;

          -- Output memory every clock cycle; this is latched in the
          -- WAITING state.
          -- Since we later use currentData as a shift register for
          -- transmission we can't output directly to currentData as
          -- this can't be inferred as RAM - it will be done with
          -- flip-flops / LUTs instead.
          memoryOutput <= memory(to_integer(readAddr));

          case sendstate is
             when WAITING =>
                tx <= '1';
```

```
            clockdiv <= 0;

            -- Latch data and start transmission
            if not bufferemptyprev then
               currentData <= memoryOutput;
               readAddr <= readAddr + 1;
               sendstate <= SEND_START;
               whichsend <= 0;
            end if;

         when SEND_START =>
            -- Pull transmit line low and wait for one bit period
            tx <= '0';
            if clockdiv = CLOCK_DIVIDER_RATIO_NORMAL then
               bitpos <= 0;
               clockdiv <= 0;
               sendstate <= SEND_DATA;

               -- If parity is being used, initialise appropriately
               if PARITY = 1 then
                  paritybit <= '1';
               elsif PARITY = 2 then
                  paritybit <= '0';
               end if;
            end if;

         when SEND_DATA =>
            -- Send current bit and wait for one bit period
            tx <= currentData(0);
            if clockdiv = CLOCK_DIVIDER_RATIO_NORMAL then
               -- Shift the data according to padding mode
               if PADDING_MODE = 1 then
                  -- Insert 1 as MSB
                  currentData <= '1' & currentData(BUFFER_WIDTH - 1
                       ...downto 1);
               elsif PADDING_MODE = 2 then
                  -- Sign extend
                  currentData <= currentData(BUFFER_WIDTH - 1) &
                       ...currentData(BUFFER_WIDTH - 1 downto 1);
               else
                  -- Insert 0 as MSB
                  currentData <= '0' & currentData(BUFFER_WIDTH - 1
                       ...downto 1);
               end if;

               -- Update parity bit if used
               if PARITY_USED then
                  if currentData(0) = '1' then
                     paritybit <= not paritybit;
                  end if;
               end if;

               -- Move on to next bit
               clockdiv <= 0;
               if bitpos = DATA_BITS - 1 then
                  -- Decide which state to go to next
                  if PARITY_USED then
                     sendstate <= SEND_PARITY;
                  else
                     sendstate <= SEND_STOP;
```

```
                    end if;
                else
                    bitpos <= bitpos + 1;
                end if;
            end if;

        when SEND_PARITY =>
            -- Send the parity bit and wait one bit period
            tx <= paritybit;
            if clockdiv = CLOCK_DIVIDER_RATIO_NORMAL then
                clockdiv <= 0;
                sendstate <= SEND_STOP;
            end if;

        when SEND_STOP =>
            -- Pull transmit line high and wait
            tx <= '1';
            if clockdiv = CLOCK_DIVIDER_RATIO_STOP then
                clockdiv <= 0;
                if whichsend = SENDS_PER_ENTRY - 1 then
                    sendstate <= WAITING;
                else
                    whichsend <= whichsend + 1;
                    sendstate <= SEND_START;
                end if;
            end if;
        end case;
    end if;
  end process senddata;
end behavioral;
```

C.3 Transmitter

```
-- transmitter: Generate the PWM waveform for the amplifier.
--
-- This module controls the class D amplifier to generate the
-- output pings. The ping rate is defined when the module is
-- instantiated. The duty cycles describing the output waveform
-- are read from external memory.
--
-- Inputs:
--   * clock: The system clock.
--   * enable: Whether the system should be pinging or not.
--   * duty: The duty cycle for the next period.
--   * endofping: If the given duty cycle is the final one of the ping.
--
-- Outputs:
--   * transmitting: The rising edge of this clock signal indicates the
--                   start of a ping.
--   * dutyclock: The rising edge of this clock indicates a new duty
--                cycle should be placed on the duty line.
--   * classd: The outputs.
--   * shutdown: Shutdown signals for the transmitters.
--
-- Version: 1.1
-- Last updated: 28 February 2010
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity transmitter is
   generic(
      CLOCK_FREQUENCY : positive := 50000000;
      PWM_FREQUENCY   : positive :=   800000;
      PING_RATE       : positive :=       10;
      NUM_CHANNELS    : positive :=        8;
      DUTY_WIDTH      : positive :=        6
   );

   port(
      clock : in std_logic;
      enable : in std_logic;
      duty : in unsigned(DUTY_WIDTH - 1 downto 0);
      endofping : in std_logic;
      transmitting : out std_logic := '0';
      dutyclock : out std_logic := '0';
      classd : out std_logic_vector(NUM_CHANNELS - 1 downto 0) :=
           ...(others => '0');
      shutdown : out std_logic_vector(NUM_CHANNELS - 1 downto 0) :=
           ...(others => '0')
   );
end transmitter;

architecture behavioral of transmitter is
   -- Clock division for ping timing
```

```
    constant PING_DIV_RATIO : natural := (CLOCK_FREQUENCY / PING_RATE) -
        ...1;
    subtype pingdiv_t is natural range 0 to PING_DIV_RATIO;
    signal pingdiv : pingdiv_t := 0;
    signal transmitbuf : std_logic := '0';

    -- Clock division to provide duty cycle clock
    constant DUTY_DIV_RATIO : natural := (CLOCK_FREQUENCY /
        ...PWM_FREQUENCY) - 1;
    constant HALF_DUTY_DIV_RATIO : natural := DUTY_DIV_RATIO / 2;
    subtype dutydiv_t is natural range 0 to DUTY_DIV_RATIO;
    signal dutydiv : dutydiv_t := 0;

    -- FSM to control module
    type state_t is (DISABLED, TRANSMIT, WAITING);
    signal state : state_t := DISABLED;

    -- Duty cycle for current period
    signal currentduty : unsigned(DUTY_WIDTH - 1 downto 0);

    -- Buffers for output signals
    signal classdbuf : std_logic := '0';
    signal shutdownbuf : std_logic := '0';
begin

-- Connect up the outputs
outputs : for index in 0 to NUM_CHANNELS - 1 generate
begin
    classd(index) <= classdbuf;
    shutdown(index) <= shutdownbuf;
end generate outputs;

doclassd : process(clock)
begin
    if clock'event and clock = '1' then
        -- Update the ping clock divider
        if pingdiv = PING_DIV_RATIO then
            pingdiv <= 0;
        else
            pingdiv <= pingdiv + 1;
        end if;

        case state is
            -- Module is disabled
            when DISABLED =>
                pingdiv <= 0;
                classdbuf <= '0';
                shutdownbuf <= '0';
                if enable = '1' then
                    state <= TRANSMIT;
                end if;

            -- Transmitting a ping
            when TRANSMIT =>
                if enable = '0' then
                    state <= DISABLED;
                else
                    -- Enable the driver IC
                    shutdownbuf <= 'Z';
                    transmitting <= '1';
```

```
                dutydiv <= dutydiv + 1;

                -- End of a period
                if dutydiv = DUTY_DIV_RATIO then
                   dutydiv <= 0;

                   -- End of the ping
                   if endofping = '1' then
                      state <= WAITING;
                   else
                      dutyclock <= '0';
                      currentduty <= duty;
                      classdbuf <= '1';
                   end if;

                -- Halfway through a period, ask for the
                -- duty cycle for next period
                elsif dutydiv = HALF_DUTY_DIV_RATIO then
                   dutyclock <= '1';
                end if;

                -- Switch the output off when appropriate
                if to_unsigned(dutydiv, DUTY_WIDTH) = currentduty then
                   classdbuf <= '0';
                end if;
             end if;

          -- Finished transmission, waiting to start next ping
          when WAITING =>
             transmitting <= '0';
             classdbuf <= '0';
             shutdownbuf <= '0';
             dutyclock <= '0';
             if enable = '0' then
                state <= DISABLED;
             elsif pingdiv = PING_DIV_RATIO then
                state <= TRANSMIT;
             end if;

       end case;
    end if;
end process doclassd;

end behavioral;
```

C.4 SPI interface

```
-- spi: Interface to the SPI controller on the AD9271.
--
-- This VHDL component provides an interface to the SPI controller on
-- the AD9271 chip. The SPI is used to configure the IC. The interface,
-- described in detail in Analog Devices application note AN-877, is
-- common to many of their high speed converters. This module should
-- work with other devices with little or no modification.
--
-- Generic parameters:
--   * INPUT_CLOCK_FREQUENCY: The frequency of the system clock.
--   * SPI_CLOCK_FREQUENCY: The desired SPI frequency. Limited to a
--                          maximum of 25MHz.
--
-- Inputs:
--   * clock: The system clock.
--   * address: The address to read or write at.
--   * datain: The data to write (if in write mode).
--   * latchin: Tells the module to latch the input and start.
--   * rnotw: If the module is reading (high) or writing (low).
--
-- Outputs:
--   * csb: The chip select bit of the SPI.
--   * sclk: The SPI clock signal.
--   * ready: If the module is ready to perform a transaction.
--   * dataout: The data read from the address.
--   * latchout: If the output data is ready.
--
-- I/O:
--   * sdio: The I/O signal of the SPI.
--
-- Version: 1.0
-- Last updated: 2 November 2009
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
library work;
use work.util.minpositive;

entity spi is
   generic(
      INPUT_CLOCK_FREQUENCY : positive := 50000000;
      SPI_CLOCK_FREQUENCY : positive := 25000000
   );

   port(
      clock : in std_logic;
      address : in std_logic_vector(7 downto 0);
      datain : in std_logic_vector(7 downto 0);
      latchin : in std_logic;
      rnotw : in std_logic;
      csb : out std_logic := '1';
      sclk : out std_logic := '1';
      sdio : inout std_logic := '0';
```

```
      ready : out std_logic := '1';
      dataout : out std_logic_vector(7 downto 0) := (others => '0');
      latchout : out std_logic := '0'
   );
end entity spi;

architecture behavioral of spi is
   -- Limit the SPI clock frequency to its maximum
   constant MAX_SPI_CLOCK_FREQUENCY : positive := 25000000;
   constant ACTUAL_SPI_CLOCK_FREQUENCY : positive :=
        work.util.minpositive(SPI_CLOCK_FREQUENCY,
        MAX_SPI_CLOCK_FREQUENCY);

   -- Clock divider to generate SPI clock
   constant CLOCK_DIV_RATIO : natural := (INPUT_CLOCK_FREQUENCY / (2 *
        ACTUAL_SPI_CLOCK_FREQUENCY)) - 1;
   subtype clockdiv_t is natural range 0 to CLOCK_DIV_RATIO;
   signal clockdiv : clockdiv_t := 0;

   -- Registers for instruction and data
   signal instruction : std_logic_vector(15 downto 0) := (others =>
        '0');
   signal senddata : std_logic_vector(7 downto 0) := (others => '0');
   signal recvdata : std_logic_vector(7 downto 0) := (others => '0');

   -- If the current instruction is sending or receiving data
   type mode_t is (SEND, RECV);
   signal mode : mode_t;

   -- States for the FSM
   type state_t is (WAITING, SEND_INSTRUCTION, SEND_DATA, RECV_DATA);
   signal state : state_t := WAITING;

   -- Counter for the number of bits sent / received in states
   subtype bitcount_t is natural range 0 to 16;
   signal bitcount : bitcount_t := 0;

   -- Buffer SCLK so its state can be used internally
   signal sclk_buf : std_logic := '1';

   -- Status signal
   signal sending : std_logic := '0';

begin

   dospi : process(clock) is
   begin
      if clock'event and clock = '1' then
         case state is
            when WAITING =>
               -- Reset output signal
               latchout <= '0';

               -- If we have information to send
               if latchin = '1' then
                  -- Put the instruction together
                  instruction(15) <= rnotw;
                  instruction(14 downto 8) <= "0000000";
                  instruction(7 downto 0) <= address;
```

```
            -- Store the send data and empty receive data
            senddata <= datain;
            recvdata <= (others => '0');

            -- Set transfer mode appropriately
            if rnotw = '1' then
               mode <= RECV;
            else
               mode <= SEND;
            end if;

            -- Next time start sending the instruction
            bitcount <= 0;
            sending <= '1';
            state <= SEND_INSTRUCTION;
         else
            sending <= '0';
         end if;

      when SEND_INSTRUCTION =>
         -- SCLK transition
         if clockdiv = CLOCK_DIV_RATIO then
            -- Falling edge, put data onto SDIO
            if sclk_buf = '1' then
               sdio <= instruction(15);
               instruction <= instruction(14 downto 0) & '0';
               bitcount <= bitcount + 1;
               sclk_buf <= '0';

            -- Rising edge, check state
            else
               sclk_buf <= '1';

               -- Sent all 16 bits
               if bitcount = 16 then
                  bitcount <= 0;

                  -- Move to the next state depending on what mode
                  -- we are in
                  if mode = RECV then
                     state <= RECV_DATA;
                  else
                     state <= SEND_DATA;
                  end if;
               end if;
            end if;
            clockdiv <= 0;

         -- Keep waiting for next transition
         else
            clockdiv <= clockdiv + 1;
         end if;

      when SEND_DATA =>
         -- SCLK transition
         if clockdiv = CLOCK_DIV_RATIO then
            -- Falling edge, put data onto SDIO
            if sclk_buf = '1' then
               sdio <= senddata(7);
               senddata <= senddata(6 downto 0) & '0';
```

```
                bitcount <= bitcount + 1;
                sclk_buf <= '0';

            -- Rising edge, check state
            else
                sclk_buf <= '1';
                if bitcount = 8 then
                    state <= WAITING;
                end if;
            end if;
            clockdiv <= 0;

        -- Keep waiting for next transition
        else
            clockdiv <= clockdiv + 1;
        end if;

    when RECV_DATA =>
        -- Put SDIO high-impedance to allow AD9271 to drive it
        sdio <= 'Z';

        -- SCLK transition
        if clockdiv = CLOCK_DIV_RATIO then
            -- Falling edge, nothing to do
            if sclk_buf = '1' then
                sclk_buf <= '0';

            -- Rising edge, shift data in
            else
                sclk_buf <= '1';

                -- Received complete octet
                if bitcount = 8 then
                    state <= WAITING;
                    dataout <= recvdata;
                    latchout <= '1';
                else
                    recvdata <= recvdata(6 downto 0) & sdio;
                    bitcount <= bitcount + 1;
                end if;
            end if;
            clockdiv <= 0;

        -- Keep waiting for next transition
        else
            clockdiv <= clockdiv + 1;
        end if;
    end case;
  end if;
end process dospi;

-- Output the SPI signals
sclk <= sclk_buf;
csb <= not sending;
ready <= not sending;

end behavioral;
```

C.5 Deserialiser

```
-- deserialiser: Convert serial input data from AD9271 to parallel.
--
-- This piece of VHDL provides a component to take the serial data
-- stream from an AD9271 channel and convert it to a parallel data
-- stream.
--
-- Inputs:
--   * dclk: The data clock from the AD9271.
--   * fclk: The frame clock from the AD9271.
--   * sdata: The serial data from the AD9271.
--   * reset: Active-high reset of the internal registers.
--
-- Outputs:
--   * data: The parallel data value.
--
-- Version 1.0
-- Last updated: 6 November 2009
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity deserialiser is
   port(
      dclk     : in std_logic;
      fclk     : in std_logic;
      sdata    : in std_logic;
      reset    : in std_logic;
      data     : out std_logic_vector(11 downto 0) := (others => '0')
   );
end deserialiser;

architecture behavioral of deserialiser is
   signal rebuffer : std_logic_vector(5 downto 0) := (others => '0');
   signal febuffer : std_logic_vector(5 downto 0) := (others => '0');
begin

   -- Sample rising edge bits
   resample : process(dclk, reset) is
   begin
      if reset = '1' then
         rebuffer <= (others => '0');
      elsif dclk'event and dclk = '1' then
         rebuffer <= rebuffer(4 downto 0) & sdata;
      end if;
   end process resample;

   -- Sample falling edge bits
   fesample : process(dclk, reset) is
   begin
      if reset = '1' then
         febuffer <= (others => '0');
      elsif dclk'event and dclk = '0' then
```

```
            febuffer <= febuffer(4 downto 0) & sdata;
        end if;
    end process fesample;

    -- Put the two buffers together on the rising edge of the frame clock
    wordsample : process(fclk, reset) is
    begin
        if reset = '1' then
            data <= (others => '0');
        elsif fclk'event and fclk = '1' then
            data(11) <= rebuffer(5);
            data(10) <= febuffer(5);
            data(9) <= rebuffer(4);
            data(8) <= febuffer(4);
            data(7) <= rebuffer(3);
            data(6) <= febuffer(3);
            data(5) <= rebuffer(2);
            data(4) <= febuffer(2);
            data(3) <= rebuffer(1);
            data(2) <= febuffer(1);
            data(1) <= rebuffer(0);
            data(0) <= febuffer(0);
        end if;
    end process wordsample;

end behavioral;
```

C.6 CIC filter

```
-- cic : VHDL implementation of a CIC decimation filter.
--
-- This VHDL component implements a generic CIC filter which can be
-- customised via the generic parameters.
--
-- Generic parameters:
--   * INPUT_WIDTH: The number of bits in the input.
--   * OUTPUT_WIDTH: The desired number of output bits.
--   * R: Decimation ratio.
--   * M: Differential delay in the comb section.
--   * N: Number of stages.
--
-- Inputs:
--   * inclock: The input (fast) clock.
--   * input: The input data.
--   * reset: Active-high reset of the internal registers.
--
-- Outputs:
--   * outclock: Output (slow) clock.
--   * output: The decimated data.
--
-- Version: 1.0
-- Last updated: 22 October 2009
-- Written by Blair Bonnett
-- Acoustics Research Group
-- Department of Electrical and Computer Engineering
-- University of Canterbury

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity cic is
   generic(
      INPUT_WIDTH  : integer := 12;
      OUTPUT_WIDTH : integer := 16;
      R            : integer := 16;
      M            : integer :=  1;
      N            : integer :=  1
   );

   port(
      inclock : in std_logic;
      input : in signed(INPUT_WIDTH - 1 downto 0);
      reset : in std_logic;
      outclock : out std_logic := '0';
      output : out signed(OUTPUT_WIDTH - 1 downto 0) := (others => '0')
   );
end cic;

architecture behavioral of cic is
   -- Required width of CIC data and assosciated type
   constant CIC_DATA_WIDTH : integer := integer(ceil(real(N) *
        ...log2(real(R) * real(M)))) + INPUT_WIDTH;
   type cicdata_t is array(natural range <>) of signed(CIC_DATA_WIDTH -
        ...1 downto 0);
```

```
   -- Integrator storage - indices are (N)(bit)
   signal cicint : cicdata_t(N - 1 downto 0) := (others => (others =>
        ...'0'));

   -- Comb storage - indices are (N)(M)(bit)
   -- Note M-element delay line plus one for output
   type ciccomb_t is array(natural range <>) of cicdata_t(M downto 0);
   signal ciccomb : ciccomb_t(N - 1 downto 0) := (others => (others =>
        ...(others => '0')));

   -- Clock divider for rate change
   subtype ratechange_t is integer range 0 to R - 1;
   constant HALFR : integer := (R / 2) - 1;
   signal ratechange : ratechange_t := 0;
   signal slowclock : std_logic := '0';

begin
   -- Create the integator chain
   integrators : for index in 0 to N - 1 generate
   begin
      integrate : process(inclock)
      begin
         if reset = '1' then
            cicint(index) <= (others => '0');
         elsif inclock'event and inclock = '1' then
            if index = 0 then
               cicint(index) <= cicint(index) + input;
            else
               cicint(index) <= cicint(index) + cicint(index - 1);
            end if;
         end if;
      end process integrate;
   end generate integrators;

   -- Create the slower clock
   reduceclock : process(inclock)
   begin
      if reset = '1' then
         slowclock <= '0';
      elsif inclock'event and inclock = '1' then
         if ratechange = R - 1 then
            ratechange <= 0;
            slowclock <= '1';
         else
            ratechange <= ratechange + 1;
         end if;

         if ratechange = HALFR then
            slowclock <= '0';
         end if;
      end if;
   end process reduceclock;

   -- Create the comb filter chain
   -- Comb storage indices are (N)(M)(bit)
   combs : for nindex in 0 to N - 1 generate
   begin
      -- Create the delay line for each comb filter
      delays : for mindex in 1 to M generate
```

```
      begin
         delay : process(slowclock)
         begin
            if reset = '1' then
               ciccomb(nindex)(mindex) <= (others => '0');
            elsif slowclock'event and slowclock = '1' then
               if mindex = 1 then
                  if nindex = 0 then
                     ciccomb(nindex)(mindex) <= cicint(N - 1);
                  else
                     ciccomb(nindex)(mindex) <= ciccomb(nindex - 1)(0);
                  end if;
               else
                  ciccomb(nindex)(mindex) <= ciccomb(nindex)(mindex -
                        ...1);
               end if;
            end if;
         end process delay;
      end generate delays;

      -- Calculate the output for each comb filter
      comb : process(slowclock)
      begin
         if reset = '1' then
            ciccomb(nindex)(0) <= (others => '0');
         elsif slowclock'event and slowclock = '1' then
            if nindex = 0 then
               ciccomb(nindex)(0) <= cicint(N - 1) - ciccomb(nindex)(M);
            else
               ciccomb(nindex)(0) <= ciccomb(nindex - 1)(0) -
                     ...ciccomb(nindex)(M);
            end if;
         end if;
      end process comb;
   end generate combs;

   -- Latch the output
   latchout : process(slowclock)
   begin
      if reset = '1' then
         output <= (others => '0');
      elsif slowclock'event and slowclock = '1' then
         output <= ciccomb(N - 1)(0);
      end if;
   end process latchout;

   -- Output the clock
   outclock <= slowclock;
end behavioral;
```

REFERENCES

Allik, H., Webman, K.M. and Hunt, J.T. (1974), 'Vibrational response of sonar transducers using piezoelectric finite elements,' *The Journal of the Acoustical Society of America*, Vol. 56, No. 6, pp. 1782–1791.

Ambardar, A. (1999), *Analog and digital signal processing*, Brooks/Cole Publishing Company, Pacific Grove, California, 2 ed.

Arnau, A. (ed.) (2004), *Piezoelectric Transducers and Applications*, Springer, Berlin.

Baker, R.J. (2008), *CMOS: circuit design, layout, and simulation*, Vol. 1, Wiley-Interscience, Hoboken, New Jersey, 2 ed.

Barclay, P.J. (2006), *Interferometric Synthetic Aperture Sonar Design and Performance*, Ph.D. thesis, University of Canterbury.

Bellanger, M.G., Bonnerot, G. and Coudreuse, M. (1976), 'Digital filtering by polyphase network: Application to sample-rate alteration and filter banks,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 24, No. 2, Apr, pp. 109–114.

Bellanger, M.G., Daguet, J.L. and Lepagnol, G.P. (1974), 'Interpolation, extrapolation, and reduction of computation speed in digital filters,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 22, No. 4, Aug, pp. 231–235.

Bowick, C. (1997), *RF Circuit Design*, Newnes, Burlington, Massachusetts, reprint; originally published by H.W. Sams, 1982.

Boylestad, R.L. and Nashelsky, L. (2002), *Electronic Devices and Circuit Theory*, Prentice Hall, New Jersey, 8 ed.

Burdic, W.S. (1984), *Underwater Acoustic System Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey.

Butterworth, S. (1914), 'On electrically-maintained vibrations,' *Proceedings of the Physical Society of London*, Vol. 27, No. 1, pp. 410–424.

Callow, H.J. (2003), *Signal Processing for Synthetic Aperture Sonar Image Enhancement*, Ph.D. thesis, University of Canterbury.

Caprais, P. and Guyonic, S. (1997), 'Squint and forward looking synthetic aperture sonar,' in *OCEANS '97. MTS/IEEE Conference Proceedings*, Vol. 2, pp. 809–814.

Coffey, M.W. (2003), 'Optimizing multistage decimation and interpolation processing,' *IEEE Signal Processing Letters*, Vol. 10, No. 4, Apr, pp. 107–110.

Cox, J.F. (2002), *Fundamentals of linear electronics : integrated and discrete*, Delmar Thomson Learning, Albany, New York, 2 ed.

Crochiere, R.E. and Rabiner, L.R. (1975), 'Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 23, No. 5, Oct, pp. 444–456.

Crochiere, R.E. and Rabiner, L.R. (1981), 'Interpolation and decimation of digital signals: a tutorial review,' *Proceedings of the IEEE*, Vol. 69, No. 3, Mar, pp. 300–331.

Crocker, M.J. (1998), *Handbook of acoustics*, Wiley Interscience, New York.

Curie, J. and Curie, P. (1880), 'Développement, par pression, de l'électricité polaire dans les cristaux hémièdres à faces inclinées,' *Comptes Rendus hebdomadaires des séances de l'Académie des Sciences*, Vol. 91, pp. 294–295.

Curie, J. and Curie, P. (1881), 'Contractions et dilatations produites par des tensions électriques dans les cristaux hémièdres à faces inclinées,' *Comptes Rendus hebdomadaires des séances de l'Académie des Sciences*, Vol. 93, pp. 1137–1140.

Dworsky, L.N. (1979), *Modern transmission line theory and applications*, Wiley, New York.

Frenzel, L.E. (1997), *Crash course in electronics technology*, Newnes, Burlington, Massachusetts, 2 ed.

Gough, P.T. and Knight, J.S. (1989), 'Wide bandwidth, constant beamwidth acoustic projectors: a simplified design procedure,' *Ultrasonics*, Vol. 27, No. 4, Jul, pp. 234–238.

Hagen, J.B. (1996), *Radio frequency circuits and systems*, Cambridge University Press, New York.

Hahnemann, W. and Hecht, H. (1920), 'Die grundform des mechanisch-akustischen schwingungskörpers,' *Physikalische Zeitschrift*, Vol. 21, pp. 187–192.

Hankel, W.G. (1881), 'Elektrische untersuchungen, fünfzehnte abhandlung über die aktino- und piezoelektrischen eigenschaften des bergkrystalles und ihre beziehung zu den thermoelektrischen,' *Abhandlungen der Königlich-Sächsischen Gesellschaft der Wissenschaften*, Vol. 12, pp. 457–548.

Harris, F.J. (2004), *Multirate signal processing for communication systems*, Prentice Hall, New Jersey.

Hawkins, D.W. and Gough, P.T. (1996), 'Multiresonance design of a Tonpilz transducer using the finite element method,' *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 43, No. 5, Sep, pp. 782–790.

Hawkins, D.W. (1996), *Synthetic Aperture Imaging Algorithms: with application to wide bandwidth sonar*, Ph.D. thesis, University of Canterbury.

Hayes, M.P. (2003), 'A real-time beam steering system for KiwiSAS-IV,' in *Proceedings of Electronics New Zealand Conference, ENZCON2003*, pp. 27–32.

Hayes, M.P. and Gough, P.T. (1992), 'Broadband synthetic aperture sonar,' *IEEE Journal of Oceanic Engineering*, Vol. 17, No. 1, Jan, pp. 80–94.

Hayes, M.P. and Gough, P.T. (2009), 'Synthetic aperture sonar: A review of current status,' *IEEE Journal of Oceanic Engineering*, Vol. 34, No. 3, Jul, pp. 207–224.

Hogenauer, E.B. (1981), 'An economical class of digital filters for decimation and interpolation,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 29, No. 2, Apr, pp. 155–162.

Hovanessian, S.A. (1980), *Introduction to synthetic array and imaging radars*, Artech House, Dedham, Massachusetts.

Hunter, A.J. (2006), *Underwater Acoustic Modelling for Synthetic Aperture Sonar*, Ph.D. thesis, University of Canterbury.

Kester, W.A. (ed.) (2005), *Data conversion handbook*, Newnes, Burlington, Massachusetts.

Knight, J.S. (1987), *Ultrasonic transducer development for a CTFM synthetic aperture sonar*, Master's thesis, University of Canterbury.

Krimholtz, R., Leedom, D.A. and Mathaei, G.L. (1970), 'New equivalent circuits for elementary piezoelectric transducers,' *Electronic Letters*, Vol. 6, No. 13, pp. 398–399.

Lerch, R., Landes, H., Friedrich, W., Hebel, R., Höss, A. and Kaarmann, H. (1992), 'Modelling of acoustic antennas with a combined finite-element-boundary-element-method,' in *IEEE 1992 Ultrasonics Symposium*, Vol. 1, pp. 581–584.

Lerch, R. and Kaarmann, H. (1987), 'Three-dimensional finite element analysis of piezoelectric media,' in *IEEE 1987 Ultrasonics Symposium*, pp. 853–858.

l'Hôpital, G.F.A.d. (1696), *Analyse des Infiniment Petits, pour l'Intelligence des Lignes Courbes*, L'Imprimerie Royale, Paris.

Lippmann, G. (1881), 'Sur le principe de la conservation de l'électricité, ou second principe de la théorie des phénomènes électriques,' *Comptes Rendus hebdomadaires des séances de l'Académie des Sciences*, Vol. 92, pp. 1049–1051.

Mason, W.P. (1948), *Electromechanical Transducers and Wave Filters*, D. Van Nostrand Company, New York, 2 ed.

Medwin, H. and Clay, C.S. (1998), *Fundamentals of acoustical oceanography*, Academic Press, Boston.

Milić, L. (2009), *Multirate Filtering for Digital Signal Processing*, Information Science Reference, London.

Mintzer, F. (1982), 'On half-band, third-band, and nth-band FIR filters and their design,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 30, No. 5, Oct, pp. 734–738.

Mueller, K.H. (1973), 'A new approach to optimum pulse shaping in sampled systems using time-domain filtering,' *The Bell System Technical Journal*, Vol. 52, May - June, pp. 723–729.

Mulligan, D.J. (2007), *An FPGA Coprocessor for Real-Time Bathymetric Synthetic Aperture Sonar*, Master's thesis, University of Canterbury.

Parks, T.W. and McClellan, J.H. (1972), 'Chebyshev approximation for nonrecursive digital filters with linear phase,' *IEEE Transactions on Circuit Theory*, Vol. 19, No. 2, Mar, pp. 189–194.

Rabiner, L.R. and Gold, B. (1975), *Theory and application of digital signal processing*, Prentice-Hall, Englewood Cliffs, New Jersey.

Rabiner, L.R. and Schafer, R.W. (1971), 'Recursive and nonrecursive realizations of digital filters designed by frequency sampling techniques,' *IEEE Transations on Audio and Electroacoustics*, Vol. 19, No. 3, Sep, pp. 200–207.

Redwood, M. (1961), 'Transient performance of a piezoelectric transducer,' *Journal of the Acoustical Society of America*, Vol. 33, No. 4, Apr, pp. 327–336.

Ritchey, L.W. (1999), 'How to design a differential signaling circuit,' *Printed Circuit Design*, Vol. 16, No. 3, Mar, pp. 14–19.

Rutledge, D.B. (1999), *The electronics of radio*, Cambridge University Press, Cambridge, UK.

Self, D. (2006), *Audio power amplifier design handbook*, Newnes, Burlington, Massachusetts, 4 ed.

Skolnik, M.I. (1980), *Introduction to radar systems*, McGraw-Hill, New York, 2 ed.

Urick, R.J. (1975), *Principles of underwater sound*, McGraw-Hill, New York, 2 ed.

Vaidyanathan, P. (1990), 'Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial,' *Proceedings of the IEEE*, Vol. 78, No. 1, Jan, pp. 56–93.

Van Dyke, K.S. (1928), 'The piezo-electric resonator and its equivalent network,' *Proceedings of the IRE*, Vol. 16, No. 6, Jun, pp. 742–764.

Zienkiewicz, O.C. and Taylor, R.L. (2000), *The Finite Element Method*, Vol. 1, Butterworth-Heinemann, Oxford, UK, 5 ed.